2020 年 4 月 19 日

# 1 [COM4513-6513] Assignment 1: Text Classification with Logistic Regression

### 1.0.1 Instructor: Nikos Aletras

The goal of this assignment is to develop and test two text classification systems:

- **Task 1:** sentiment analysis, in particular to predict the sentiment of movie review, i.e. positive or negative (binary classification).
- **Task 2:** topic classification, to predict whether a news article is about International issues, Sports or Business (multiclass classification).

For that purpose, you will implement:

- Text processing methods for extracting Bag-Of-Word features, using (1) unigrams, bigrams and trigrams to obtain vector representations of documents. Two vector weighting schemes should be tested: (1) raw frequencies (**3 marks; 1 for each ngram type**); (2) tf.idf (**1 marks**).
- Binary Logistic Regression classifiers that will be able to accurately classify movie reviews trained with (1) BOW-count (raw frequencies); and (2) BOW-tfidf (tf.idf weighted) for Task 1.
- Multiclass Logistic Regression classifiers that will be able to accurately classify news articles trained with (1) BOW-count (raw frequencies); and (2) BOW-tfidf (tf.idf weighted) for Task 2.
- The Stochastic Gradient Descent (SGD) algorithm to estimate the parameters of your Logistic Regression models. Your SGD algorithm should:
  - Minimise the Binary Cross-entropy loss function for Task 1 (**3 marks**)
  - Minimise the Categorical Cross-entropy loss function for Task 2 (**3 marks**)
  - Use L2 regularisation (both tasks) (**1 mark**)

  - Perform multiple passes (epochs) over the training data (**1 mark**)
  - Randomise the order of training data after each pass (**1 mark**)
  - Stop training if the difference between the current and previous validation loss is smaller than a threshold (**1 mark**)
  - After each epoch print the training and development loss (**1 mark**)
- Discuss how did you choose hyperparameters (e.g. learning rate and regularisation strength)? (**2 marks; 0.5 for each model in each task**).
- After training the LR models, plot the learning process (i.e. training and validation loss in each epoch) using a line plot (**1 mark; 0.5 for both BOW-count and BOW-tfidf LR models in each task**) and discuss if your model overfits/underfits/is about right.
- Model interpretability by showing the most important features for each class (i.e. most positive/negative weights). Give the top 10 for each class and comment on whether they make sense (if they don' t you might have a bug!). If we were to apply the classifier we' ve learned into a different domain such laptop reviews or restaurant reviews, do you think these features would generalise well? Can you propose what features the classifier could pick up as important in the new domain? (**2 marks; 0.5 for BOW-count and BOW-tfidf LR models respectively in each task**)

### 1.0.2 Data - Task 1

The data you will use for Task 1 are taken from here: `http://www.cs.cornell.edu/people/pabo/movie-review-data/` and you can find it in the `./data_sentiment` folder in CSV format:

- `data_sentiment/train.csv`: contains 1,400 reviews, 700 positive (label: 1) and 700 negative (label: 0) to be used for training.
- `data_sentiment/dev.csv`: contains 200 reviews, 100 positive and 100 negative to be used for hyperparameter selection and monitoring the training process.
- `data_sentiment/test.csv`: contains 400 reviews, 200 positive and 200 negative to be used for testing.

### 1.0.3 Data - Task 2

The data you will use for Task 2 is a subset of the AG News Corpus and you can find it in the `./data_topic` folder in CSV format:

- `data_topic/train.csv`: contains 2,400 news articles, 800 for each class to be used for training.
- `data_topic/dev.csv`: contains 150 news articles, 50 for each class to be used for hyperparameter selection and monitoring the training process.

- `data_topic/test.csv`: contains 900 news articles, 300 for each class to be used for testing.

### 1.0.4 Submission Instructions

You should submit a Jupyter Notebook file (assignment1.ipynb) and an exported PDF version (you can do it from Jupyter: `File->Download as->PDF via Latex`).

You are advised to follow the code structure given in this notebook by completing all given funtions. You can also write any auxilliary/helper functions (and arguments for the functions) that you might need but note that you can provide a full solution without any such functions. Similarly, you can just use only the packages imported below but you are free to use any functionality from the Python Standard Library, NumPy, SciPy and Pandas. You are not allowed to use any third-party library such as Scikit-learn (apart from metric functions already provided), NLTK, Spacy, Keras etc..

Please make sure to comment your code. You should also mention if you’ ve used Windows (not recommended) to write and test your code. There is no single correct answer on what your accuracy should be, but correct implementations usually achieve F1-scores around 80% or higher. The quality of the analysis of the results is as important as the accuracy itself.

This assignment will be marked out of 20. It is worth 20% of your final grade in the module.

The deadline for this assignment is **23:59 on Fri, 20 Mar 2020** and it needs to be submitted via MOLE. Standard departmental penalties for lateness will be applied. We use a range of strategies to detect unfair means, including Turnitin which helps detect plagiarism, so make sure you do not plagiarise.

[1]:

```
import pandas as pd
import numpy as np
from collections import Counter
import re
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
import random
import math
```

```
# fixing random seed for reproducibility
random.seed(123)
np.random.seed(123)
```

## 1.1 Load Raw texts and labels into arrays

First, you need to load the training, development and test sets from their corresponding CSV files (tip: you can use Pandas dataframes).

[2]:
```
# fill in your code...
data_dev = pd.read_csv("./data_sentiment/dev.csv",header=None,␣
 ↪names=['text','label'])
data_test = pd.read_csv("./data_sentiment/test.csv",header=None,␣
 ↪names=['text','label'])
data_tr = pd.read_csv("./data_sentiment/train.csv",header=None,␣
 ↪names=['text','label'])
```

[3]:
```
type(data_tr)
```

[3]:
```
pandas.core.frame.DataFrame
```

If you use Pandas you can see a sample of the data.

[4]:
```
data_tr.head()
```

[4]:
```
                                                text  label
0  note : some may consider portions of the follo…      1
1  note : some may consider portions of the follo…      1
2  every once in a while you see a film that is s…      1
3  when i was growing up in 1970s , boys in my sc…      1
4  the muppet movie is the first , and the best m…      1
```

The next step is to put the raw texts into Python lists and their corresponding labels into NumPy arrays:

[5]:
```
# Put the raw texts into list
# training dataset
data_tr_list = data_tr['text'].tolist()
data_tr_label = np.array(data_tr['label'])
```

```
# development dataset
data_dev_list = data_dev['text'].tolist()
data_dev_label = np.array(data_dev['label'])
# test dataset
data_test_list = data_test['text'].tolist()
data_test_label = np.array(data_test['label'])
```

# 2 Bag-of-Words Representation

To train and test Logisitc Regression models, you first need to obtain vector representations for all documents given a vocabulary of features (unigrams, bigrams, trigrams).

## 2.1 Text Pre-Processing Pipeline

To obtain a vocabulary of features, you should: - tokenise all texts into a list of unigrams (tip: using a regular expression) - remove stop words (using the one provided or one of your preference) - compute bigrams, trigrams given the remaining unigrams - remove ngrams appearing in less than K documents - use the remaining to create a vocabulary of unigrams, bigrams and trigrams (you can keep top N if you encounter memory issues).

[6]:

```
stop_words = ['a','in','on','at','and','or',
              'to', 'the', 'of', 'an', 'by',
              'as', 'is', 'was', 'were', 'been', 'be',
              'are','for', 'this', 'that', 'these', 'those', 'you', 'i',
             'it', 'he', 'she', 'we', 'they' 'will', 'have', 'has',
              'do', 'did', 'can', 'could', 'who', 'which', 'what',
             'his', 'her', 'they', 'them', 'from', 'with', 'its']
```

### 2.1.1 N-gram extraction from a document

You first need to implement the `extract_ngrams` function. It takes as input: - `x_raw`: a string corresponding to the raw text of a document - `ngram_range`: a tuple of two integers denoting the type of ngrams you want to extract, e.g. (1,2) denotes extracting unigrams and bigrams. - `token_pattern`: a string to be used within a regular expression to extract all tokens. Note that data is already tokenised so you could opt for a simple white space tokenisation. - `stop_words`: a list of stop words - `vocab`: a given vocabulary. It should be used to extract specific features.

and returns:

- a list of all extracted features.

See the examples below to see how this function should work.

[7]:

```
def extract_ngrams(x_raw, ngram_range=(1,3),␣
 ↪token_pattern=r'\b[A-Za-z][A-Za-z]+\b', stop_words=[], vocab=set()):
    # Next it is my part to finish the this part
    # Create a new blank list
    x = list()
    ngram_list = list()
    # judge the variable type
    if type(x_raw) == type("string"):
        x_raw = list([x_raw])

    for x_each in x_raw:
        x_findword = re.findall(token_pattern,x_each)
        # Now we have obtained the new word list without the stop_word
        x_rawall = [word for word in x_findword if word not in stop_words]
        store_ngram = list()
        for n_g in range(min(ngram_range),max(ngram_range)+1):
            for i in range(len(x_rawall)-n_g+1):
                if n_g == 1:
                    ngramTemp = x_rawall[i]
                    x.append(ngramTemp)
                    store_ngram.append(ngramTemp)
                elif n_g == 2 or n_g == 3:
                    ngramTemp = tuple(x_rawall[i:i+n_g])
                    x.append(ngramTemp)
                    store_ngram.append(ngramTemp)
        ngram_list.append(store_ngram)
    # Now we should make the x into a set and
    if vocab:
        x = set(x) & vocab
        return x
    else:
        return ngram_list
```

```
[8]: extract_ngrams("this is a great movie to watch",
                    ngram_range=(1,3),
                    stop_words=stop_words)
```

```
[8]: [['great',
       'movie',
       'watch',
       ('great', 'movie'),
       ('movie', 'watch'),
       ('great', 'movie', 'watch')]]
```

```
[9]: extract_ngrams("this is a great movie to watch",
                    ngram_range=(1,2),
                    stop_words=stop_words,
                    vocab=set(['great',  ('great','movie')]))
```

```
[9]: {('great', 'movie'), 'great'}
```

Note that it is OK to represent n-grams using lists instead of tuples: e.g. `['great', ['great', 'movie']]`

### 2.1.2 Create a vocabulary of n-grams

Then the `get_vocab` function will be used to (1) create a vocabulary of ngrams; (2) count the document frequencies of ngrams; (3) their raw frequency. It takes as input: - `X_raw`: a list of strings each corresponding to the raw text of a document - `ngram_range`: a tuple of two integers denoting the type of ngrams you want to extract, e.g. (1,2) denotes extracting unigrams and bigrams. - `token_pattern`: a string to be used within a regular expression to extract all tokens. Note that data is already tokenised so you could opt for a simple white space tokenisation. - `stop_words`: a list of stop words - `vocab`: a given vocabulary. It should be used to extract specific features. - `min_df`: keep ngrams with a minimum document frequency. - `keep_topN`: keep top-N more frequent ngrams.

and returns:

- `vocab`: a set of the n-grams that will be used as features.
- `df`: a Counter (or dict) that contains ngrams as keys and their corresponding document frequency as values.
- `ngram_counts`: counts of each ngram in vocab

Hint: it should make use of the `extract_ngrams` function.

[10]:
```
def get_vocab(X_raw, ngram_range=(1,3), token_pattern=r'\b[A-Za-z][A-Za-z]+\b', min_df=0, keep_topN=0, stop_words=[]):

    # Firstly we should obtain the all the ngram list
    # From the previous function this function will choose the putput type due to the input variables
    vocab = extract_ngrams(x_raw=X_raw, stop_words=stop_words)

    df = dict() #dict that contains ngrams as keys and their corresponding document frequency as values.
    ngram_counts= list() # type is the list and stores the number of words which occur at the text

    for each_ngram in vocab:
        for ngram in set(each_ngram):
            if ngram not in df.keys():
                df[ngram] = 1
            else:
                df[ngram] +=1

    df = dict(sorted(df.items(), key=lambda count:count[1], reverse=True)[:keep_topN])
    vocab = [name for name in df.keys()]

    for item in df.keys():
        ngram_counts.append(item)

    return vocab, df, ngram_counts
```

Now you should use `get_vocab` to create your vocabulary and get document and raw frequencies of n-grams:

[11]:
```
X_tr_raw = data_tr_list
vocab, df, ngram_counts = get_vocab(X_tr_raw, ngram_range=(1,3), keep_topN=5000, stop_words=stop_words)
```

```
print(len(vocab))
print()
print(list(vocab)[:100])
print()
print(list(df.items())[:10])
# print(df.most_common()[:10])
```

```
5000

['but', 'one', 'film', 'not', 'all', 'movie', 'out', 'so', 'there', 'like',
'more', 'up', 'about', 'when', 'some', 'if', 'just', 'only', 'into', 'than',
'even', 'their', 'time', 'most', 'no', 'good', 'much', 'him', 'would', 'other',
'get', 'story', 'well', 'will', 'also', 'two', 'after', 'first', 'character',
'make', 'way', 'characters', 'off', 'see', 'very', 'while', 'does', 'any',
'where', 'too', 'little', 'plot', 'because', 'over', 'director', 'had', 'how',
'then', 'best', 'being', 'people', 'doesn', 'really', 'man', 'never', 'life',
'through', 'films', 'here', 'don', 'many', 'another', 'such', 'scene', 'me',
'bad', 'know', 'made', 'scenes', 'my', 'end', 'new', 'go', 'before', 'back',
'makes', 'something', 'great', 'work', 'movies', 'still', 'better', 'now',
'few', 'down', 'seems', 'around', 'every', 're', 'enough']

[('but', 1334), ('one', 1247), ('film', 1231), ('not', 1170), ('all', 1117),
('movie', 1095), ('out', 1080), ('so', 1047), ('there', 1046), ('like', 1043)]
```

Then, you need to create vocabulary id -> word and id -> word dictionaries for reference:

[12]:

```
# fill in your code...
# We need to number the each words and store them in a dictionary
voca_dict = dict()
for i in range(len(vocab)):
    voca_dict[i] = vocab[i]


print(list(voca_dict.items())[:5])
```

```
[(0, 'but'), (1, 'one'), (2, 'film'), (3, 'not'), (4, 'all')]
```

Now you should be able to extract n-grams for each text in the training, development and test sets:

[13]:

```
# Finish the process of extracting the n-grams for feature
tr_feature, tr_df, tr_ngramcount = get_vocab(data_tr_list,ngram_range=(1,3),␣
 ↪keep_topN=5000, stop_words=stop_words)
dev_feature, dev_df, dev_ngramcount =␣
 ↪get_vocab(data_dev_list,ngram_range=(1,3), keep_topN=5000,␣
 ↪stop_words=stop_words)
test_feature, test_df, test_ngramcount =␣
 ↪get_vocab(data_test_list,ngram_range=(1,3), keep_topN=5000,␣
 ↪stop_words=stop_words)

# The part of extracting the ngram from the extract_ngrams
tr_ngram = extract_ngrams(data_tr_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
dev_ngram = extract_ngrams(data_dev_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
test_ngram = extract_ngrams(data_test_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
```

## 2.2 Vectorise documents

Next, write a function `vectoriser` to obtain Bag-of-ngram representations for a list of documents. The function should take as input: - `X_ngram`: a list of texts (documents), where each text is represented as list of n-grams in the `vocab` - `vocab`: a set of n-grams to be used for representing the documents

and return: - `X_vec`: an array with dimensionality Nx|vocab| where N is the number of documents and |vocab| is the size of the vocabulary. Each element of the array should represent the frequency of a given n-gram in a document.

[14]:

```
def vectorise(X_ngram, vocab):
    # Now we have obtain all the ngram and the type is list
    # Use the nparray to establish a new matrix
    X_matrix = np.zeros((len(X_ngram),len(vocab)))
    for sid,ngram in enumerate(X_ngram):
        for sid_v,ngram_top in enumerate(vocab):
            X_matrix[sid][sid_v] = ngram.count(ngram_top)
    return X_matrix
```

Finally, use `vectorise` to obtain document vectors for each document in the train, development and test set. You should extract both count and tf.idf vectors respectively:

**Count vectors**

```
[15]: # fill in your code...
      X_tr_count = vectorise(tr_ngram,tr_feature)
      X_dev_count = vectorise(dev_ngram,tr_feature)
      X_test_count = vectorise(test_ngram,tr_feature)
```

```
[16]: X_tr_count[:2,:50]
```

```
[16]: array([[ 6.,  8., 20.,  4.,  1.,  0.,  1.,  3.,  1.,  0.,  1.,  0.,  1.,
               0.,  1.,  1.,  2.,  4.,  2.,  1.,  3.,  6.,  0.,  4.,  1.,  1.,
               1.,  0.,  3.,  0.,  0.,  2.,  0.,  1.,  0.,  1.,  0.,  2.,  3.,
               0.,  0.,  4.,  1.,  1.,  0.,  3.,  0.,  1.,  1.,  0.],
             [ 2.,  5.,  6.,  2.,  4.,  0.,  2.,  3.,  2.,  3.,  3.,  4.,  2.,
               0.,  2.,  2.,  0.,  0.,  2.,  3.,  0.,  0.,  2.,  2.,  1.,  1.,
               1.,  5.,  1.,  1.,  1.,  2.,  0.,  4.,  1.,  1.,  0.,  0.,  5.,
               1.,  2.,  0.,  0.,  1.,  0.,  3.,  1.,  1.,  2.,  0.]])
```

```
[17]: X_tr_count.shape
```

```
[17]: (1400, 5000)
```

**TF.IDF vectors** First compute `idfs` an array containing inverted document frequencies (Note: its elements should correspond to your `vocab`)

```
[18]: def idf_function(ngram_list,vocab):

          # establish a new dictionary datastructure to store the idf value
          idf_dict = dict()
          num_N = len(ngram_list)
          for sid,voc in enumerate(vocab):
              occur_sum = 0
              for ngram in ngram_list:
                  if voc in ngram:
                      occur_sum +=1
              idf_dict[voc]= np.log((num_N+1)/(occur_sum+1))
```

```
    return idf_dict
```

Then transform your count vectors to tf.idf vectors:

```
[19]: # Now we have obtained the dictionary about the idf value
      # X_tr_count - the original vector matrix
      # x_raw - the raw data
      def tfidf_function(X_count,ngram_list,vocab):
          tfidf = list()
          # obtain the new idf_dict
          idf_dict = idf_function(ngram_list,vocab)
          for sid,word in enumerate(idf_dict.keys()):
              result = X_count[:,sid] * idf_dict[word]
              tfidf.append(result)

          X_tfidf = np.transpose(np.asarray(tfidf))

          return X_tfidf
```

```
[20]: X_tr_tfidf = tfidf_function(X_count=X_tr_count, ngram_list = tr_ngram, vocab =␣
       ↪tr_feature)
      X_dev_tfidf = tfidf_function(X_count=X_dev_count, ngram_list = dev_ngram, vocab␣
       ↪= tr_feature)
      X_test_tfidf = tfidf_function(X_count=X_test_count, ngram_list = test_ngram,␣
       ↪vocab = tr_feature)
```

```
[21]: X_tr_tfidf[1,:50]
```

```
[21]: array([0.09650995, 0.57821999, 0.77128441, 0.35865637, 0.90257957,
             0.        , 0.51859946, 0.87090804, 0.58251467, 0.88238033,
             0.89101343, 1.26561491, 0.65249265, 0.        , 0.74355542,
             0.79812334, 0.        , 0.        , 0.83896302, 1.284617  ,
             0.        , 0.        , 0.94143532, 0.99471004, 0.50442219,
             0.52834677, 0.55658683, 2.864543  , 0.5895012 , 0.59336967,
             0.59985058, 1.20230363, 0.        , 2.45175908, 0.61690017,
             0.62087632, 0.        , 0.        , 3.16450256, 0.68108602,
             1.36499491, 0.        , 0.        , 0.70535559, 0.        ,
       2.17296126, 0.74365188, 0.74515451, 1.50843602, 0.        ])
```

## 3 Binary Logistic Regression

After obtaining vector representations of the data, now you are ready to implement Binary Logistic Regression for classifying sentiment.

First, you need to implement the `sigmoid` function. It takes as input:

- `z`: a real number or an array of real numbers

and returns:

- `sig`: the sigmoid of `z`

[22]:
```python
def sigmoid(z):
    z = 1.0 / (1 + np.exp(-z))
    return z
```

[23]:
```python
print(sigmoid(0))
print(sigmoid(np.array([-5., 1.2])))
```

```
0.5
[0.00669285 0.76852478]
```

Then, implement the `predict_proba` function to obtain prediction probabilities. It takes as input:

- `X`: an array of inputs, i.e. documents represented by bag-of-ngram vectors ($N \times |vocab|$)
- `weights`: a 1-D array of the model’ s weights $(1, |vocab|)$

and returns:

- `preds_proba`: the prediction probabilities of X given the weights

[24]:
```python
def predict_proba(X, weights):

    # Calculate inner product of the matrix
    preds_proba = sigmoid(np.dot(X,weights))

    return preds_proba
```

Then, implement the `predict_class` function to obtain the most probable class for each vector in an array of input vectors. It takes as input:

- `X`: an array of documents represented by bag-of-ngram vectors ($N \times |vocab|$)
- `weights`: a 1-D array of the model’ s weights $(1, |vocab|)$

and returns:

- `preds_class`: the predicted class for each x in X given the weights

[25]:
```python
def predict_class(X, weights):

    # Calculate the prediction probabilities of X given the weights
    proba_value = predict_proba(X,weights)
    # Change the value into 1 or 0
    proba_value = np.where(proba_value<0.5, 0, 1)
    preds_class = proba_value.astype(np.int64)

    return preds_class
```

To learn the weights from data, we need to minimise the binary cross-entropy loss. Implement `binary_loss` that takes as input:

- `X`: input vectors
- `Y`: labels
- `weights`: model weights
- `alpha`: regularisation strength

and return:

- `l`: the loss score

[26]:
```python
def binary_loss(X, Y, weights, alpha=0.00001):

    # Firstly we should calculate the prediction probabilities of X used with weight
    y_proba = predict_proba(X,weights)

    # Restrict the value range from the alpha  to  1-alpha
    y_proba  = np.clip(y_proba,alpha,1-alpha)
    loss = - Y * np.log(y_proba) - (1 - Y) * np.log(1-y_proba)
    # Look the X scope
    if len(X.shape)>1:
        L2_regularization = (1/len(X))*(alpha/2)*(np.sum(np.square(weights)))
    else:
        L2_regularization = (alpha/2)*(np.sum(np.square(weights)))

    l = loss + L2_regularization
    return l
```

Now, you can implement Stochastic Gradient Descent to learn the weights of your sentiment classifier. The `SGD` function takes as input:

- `X_tr`: array of training data (vectors)
- `Y_tr`: labels of `X_tr`
- `X_dev`: array of development (i.e. validation) data (vectors)
- `Y_dev`: labels of `X_dev`
- `lr`: learning rate
- `alpha`: regularisation strength
- `epochs`: number of full passes over the training data
- `tolerance`: stop training if the difference between the current and previous validation loss is smaller than a threshold
- `print_progress`: flag for printing the training progress (train/validation loss)

and returns:

- `weights`: the weights learned
- `training_loss_history`: an array with the average losses of the whole training set after each epoch
- `validation_loss_history`: an array with the average losses of the whole development set after each epoch

[27]:

```
def SGD(X_tr, Y_tr, X_dev=[], Y_dev=[], loss="binary", lr=0.1, alpha=0.00001,␣
↪epochs=5, tolerance=0.0001, print_progress=True):

    cur_loss_dev = 1.
    training_loss_history = []
    validation_loss_history = []

    # Obtain the weights
    weights = np.zeros(X_tr.shape[1])

    for i in range(epochs):
        loss_list = list()
        seed_number = random.randint(0,100)
        np.random.seed(seed_number)
        # Shuffle the X_tr and Y_tr
        per_X_tr = np.random.permutation(X_tr)
        np.random.seed(seed_number)
        per_Y_tr = np.random.permutation(Y_tr)
        for j,row in enumerate(per_X_tr):
            # Caculate the Binary Loss and store the loss into loss_list
            loss_tr = binary_loss(row, per_Y_tr[j],weights, alpha)
            loss_list.append(loss_tr)
            y_pred = predict_proba(row,weights)
            error = y_pred - per_Y_tr[j]
            # update weights
            weights = weights - lr*error*row
        # Obtain the mean
        tr_loss_mean = np.mean(loss_list)
        training_loss_history.append(tr_loss_mean)

        dev_loss = binary_loss(X_dev, Y_dev, weights, alpha)
        dev_loss = sum(dev_loss)/len(dev_loss)
        validation_loss_history.append(dev_loss)
        print('Epoch: %d' % i, '| Training loss: %f' %tr_loss_mean, '| Validation loss: %f' %dev_loss)

        if (cur_loss_dev-dev_loss)<tolerance:
            break

        cur_loss_dev = dev_loss
    return weights, training_loss_history, validation_loss_history
```

## 3.1 Train and Evaluate Logistic Regression with Count vectors

First train the model using SGD:

[28]:
```
w_count, loss_tr_count, dev_loss_count = SGD(X_tr_count, data_tr_label,
                                             X_dev=X_dev_count,
                                             Y_dev=data_dev_label,
                                             lr=0.0001,
                                             alpha=0.001,
                                             epochs=100)
```

```
Epoch: 0 | Training loss: 0.667690 | Validation loss: 0.647094
Epoch: 1 | Training loss: 0.613934 | Validation loss: 0.615373
Epoch: 2 | Training loss: 0.575116 | Validation loss: 0.595635
Epoch: 3 | Training loss: 0.545252 | Validation loss: 0.575331
Epoch: 4 | Training loss: 0.520551 | Validation loss: 0.562030
Epoch: 5 | Training loss: 0.499586 | Validation loss: 0.548129
Epoch: 6 | Training loss: 0.481713 | Validation loss: 0.538025
Epoch: 7 | Training loss: 0.465533 | Validation loss: 0.530481
Epoch: 8 | Training loss: 0.451564 | Validation loss: 0.519784
Epoch: 9 | Training loss: 0.439090 | Validation loss: 0.512905
Epoch: 10 | Training loss: 0.427257 | Validation loss: 0.506482
Epoch: 11 | Training loss: 0.416128 | Validation loss: 0.501852
Epoch: 12 | Training loss: 0.406814 | Validation loss: 0.494111
Epoch: 13 | Training loss: 0.397518 | Validation loss: 0.488875
Epoch: 14 | Training loss: 0.388979 | Validation loss: 0.484013
Epoch: 15 | Training loss: 0.380997 | Validation loss: 0.480896
Epoch: 16 | Training loss: 0.373599 | Validation loss: 0.475360
Epoch: 17 | Training loss: 0.366493 | Validation loss: 0.471957
Epoch: 18 | Training loss: 0.359441 | Validation loss: 0.469523
Epoch: 19 | Training loss: 0.353198 | Validation loss: 0.465293
Epoch: 20 | Training loss: 0.347355 | Validation loss: 0.462088
Epoch: 21 | Training loss: 0.341564 | Validation loss: 0.457970
Epoch: 22 | Training loss: 0.336115 | Validation loss: 0.455706
Epoch: 23 | Training loss: 0.330757 | Validation loss: 0.453449
```

```
Epoch: 24 | Training loss: 0.325950 | Validation loss: 0.450681
Epoch: 25 | Training loss: 0.320996 | Validation loss: 0.448717
Epoch: 26 | Training loss: 0.316457 | Validation loss: 0.445406
Epoch: 27 | Training loss: 0.311947 | Validation loss: 0.442609
Epoch: 28 | Training loss: 0.307677 | Validation loss: 0.441136
Epoch: 29 | Training loss: 0.303613 | Validation loss: 0.439073
Epoch: 30 | Training loss: 0.299752 | Validation loss: 0.436814
Epoch: 31 | Training loss: 0.295824 | Validation loss: 0.434723
Epoch: 32 | Training loss: 0.291876 | Validation loss: 0.433425
Epoch: 33 | Training loss: 0.288540 | Validation loss: 0.431482
Epoch: 34 | Training loss: 0.285090 | Validation loss: 0.429750
Epoch: 35 | Training loss: 0.281504 | Validation loss: 0.429179
Epoch: 36 | Training loss: 0.278308 | Validation loss: 0.426556
Epoch: 37 | Training loss: 0.275335 | Validation loss: 0.425370
Epoch: 38 | Training loss: 0.272124 | Validation loss: 0.424008
Epoch: 39 | Training loss: 0.269183 | Validation loss: 0.423003
Epoch: 40 | Training loss: 0.266316 | Validation loss: 0.421179
Epoch: 41 | Training loss: 0.263219 | Validation loss: 0.420743
Epoch: 42 | Training loss: 0.260617 | Validation loss: 0.418752
Epoch: 43 | Training loss: 0.257886 | Validation loss: 0.417834
Epoch: 44 | Training loss: 0.255196 | Validation loss: 0.416404
Epoch: 45 | Training loss: 0.252851 | Validation loss: 0.415398
Epoch: 46 | Training loss: 0.250213 | Validation loss: 0.415025
Epoch: 47 | Training loss: 0.247883 | Validation loss: 0.413822
Epoch: 48 | Training loss: 0.245570 | Validation loss: 0.412777
Epoch: 49 | Training loss: 0.243239 | Validation loss: 0.411473
Epoch: 50 | Training loss: 0.240952 | Validation loss: 0.410816
Epoch: 51 | Training loss: 0.238663 | Validation loss: 0.409894
Epoch: 52 | Training loss: 0.236534 | Validation loss: 0.408841
Epoch: 53 | Training loss: 0.234449 | Validation loss: 0.408064
Epoch: 54 | Training loss: 0.232368 | Validation loss: 0.407237
Epoch: 55 | Training loss: 0.230297 | Validation loss: 0.406769
Epoch: 56 | Training loss: 0.228253 | Validation loss: 0.406385
Epoch: 57 | Training loss: 0.226466 | Validation loss: 0.405153
Epoch: 58 | Training loss: 0.224542 | Validation loss: 0.404563
Epoch: 59 | Training loss: 0.222540 | Validation loss: 0.403616
Epoch: 60 | Training loss: 0.220682 | Validation loss: 0.403421
```

```
Epoch: 61 | Training loss: 0.218944 | Validation loss: 0.403166
Epoch: 62 | Training loss: 0.217157 | Validation loss: 0.401730
Epoch: 63 | Training loss: 0.215344 | Validation loss: 0.401756
```

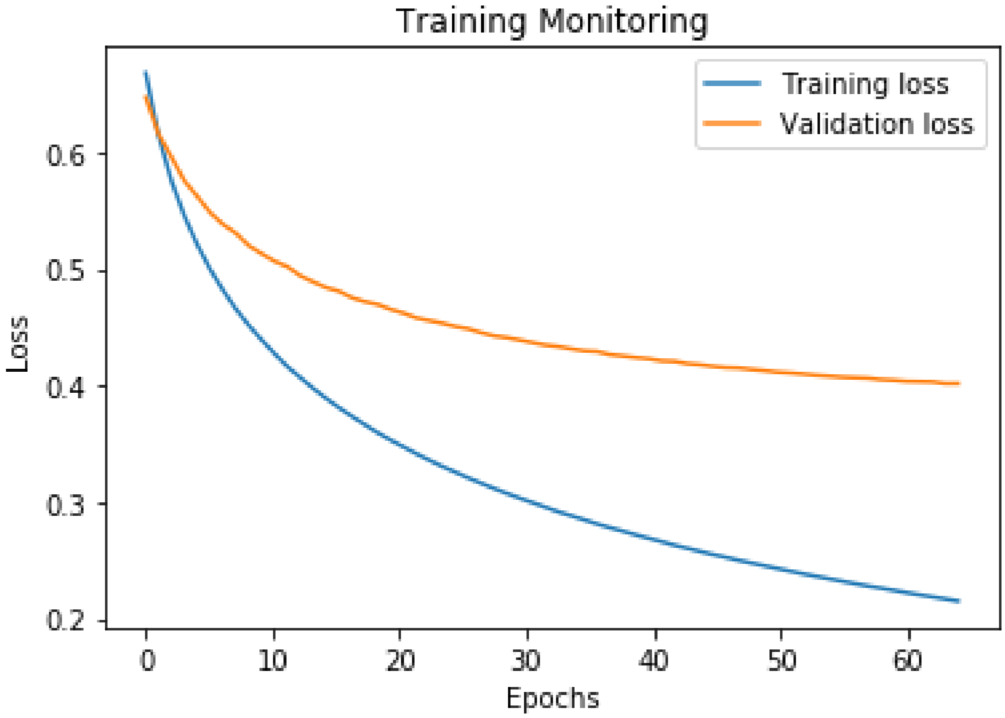
Now plot the training and validation history per epoch. Does your model underfit, overfit or is it about right? Explain why.

[29]:

```
x = np.linspace(0,len(loss_tr_count),len(loss_tr_count))

plt.plot(x, loss_tr_count,label='Training loss')
plt.plot(x, dev_loss_count, label='Validation loss')

plt.title('Training Monitoring')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Explain here...

**Answer**:

The model have reached at the overfit situation,because the gap between the Training loss and Validation loss is very large. The over-training data result in optimization. In addition, the model' s generalization ability performs not well. These conditions cause the error rate of verification set is higher than the traininng set.

Compute accuracy, precision, recall and F1-scores:

[30]:

```
# fill in your code...
def accuracy_score(Y_trainerror,pre_te):
    result = 0
    for i,label in enumerate(Y_trainerror):
        if pre_te[i] == label:
            result = result+1
    score = result/len(Y_trainerror)
    return score

def precision_score(Y_trainerror,pre_te):
    Y_cor = (Y_trainerror==1)
    pre_te_cor = (pre_te==1)
    cor_count = (Y_cor * pre_te_cor).sum()
    all_count = (pre_te==1).sum()

    return cor_count/all_count

def recall_score(Y_trainerror,pre_te):
    Y_cor = (Y_trainerror==1)
    pre_te_cor = (pre_te==1)
    cor_count = (Y_cor * pre_te_cor).sum()
    all_count = (Y_cor==1).sum()

    return cor_count/all_count

def f1_score(Y_trainerror,pre_te):
    cor_count = 2 * precision_score(Y_trainerror, pre_te) *␣
 ↪recall_score(Y_trainerror, pre_te)
```

```
    all_count = precision_score(Y_trainerror, pre_te) +␣
↪recall_score(Y_trainerror, pre_te)

    return cor_count/all_count
```

[31]:

```
preds_te_count = predict_class(X_test_count, w_count)
Y_te = data_test_label

print('Accuracy:', accuracy_score(Y_te,preds_te_count))
print('Precision:', precision_score(Y_te,preds_te_count))
print('Recall:', recall_score(Y_te,preds_te_count))
print('F1-Score:', f1_score(Y_te,preds_te_count))
```

```
Accuracy: 0.8325
Precision: 0.824390243902439
Recall: 0.845
F1-Score: 0.8345679012345678
```

Finally, print the top-10 words for the negative and positive class respectively.

[32]:

```
# fill in your code...
# Obtain a new dictionary to store the weight
dic_weight = dict()
for i,weight in enumerate(w_count):
    dic_weight[vocab[i]] = weight

# Obtain the positive words
dic_resultP = dict()
dic_resultN = dict()

for word,weight in dic_weight.items():
    if weight >= 0:
        dic_resultP[word] = weight
    else:
        dic_resultN[word] = weight
```

```
dic_resultP = dict(sorted(dic_resultP.items(),key=lambda weight:weight[1],␣
 ↪reverse=True)[:10])
dic_resultN = dict(sorted(dic_resultN.items(),key=lambda weight:weight[1],␣
 ↪reverse=False)[:10])

print("The top 10 positive words are %s respectively."% list(dic_resultP.
 ↪keys()))
print("The top 10 negative words are %s respectively."% list(dic_resultN.
 ↪keys()))
```

```
The top 10 positive words are ['great', 'well', 'also', 'seen', 'life', 'fun',
'world', 'many', 'movies', 'see'] respectively.
The top 10 negative words are ['bad', 'only', 'worst', 'unfortunately',
'script', 'why', 'plot', 'boring', 'any', 'nothing'] respectively.
```

If we were to apply the classifier we’ ve learned into a different domain such laptop reviews or restaurant reviews, do you think these features would generalise well? Can you propose what features the classifier could pick up as important in the new domain?

Provide your answer here…

**Answer**:

We could use this trained features into the different domain such as the shopping reviews or restaurant reviews.

The features like positive category such as ‘well’ ‘good’ and negative category such as ‘nothing’ ‘worst’ ‘unfortunately’ .

But we should remove some irrelevant words from the data set.

### 3.2 Train and Evaluate Logistic Regression with TF.IDF vectors

Follow the same steps as above (i.e. evaluating count n-gram representations).

```
[33]: w_tfidf, trl, devl = SGD(X_tr_tfidf, data_tr_label,
                             X_dev=X_dev_tfidf,
                             Y_dev=data_dev_label,
                             lr=0.0001,
                             alpha=0.00001,
                         epochs=50)
```

```
Epoch: 0 | Training loss: 0.630729 | Validation loss: 0.597229
Epoch: 1 | Training loss: 0.493460 | Validation loss: 0.549631
Epoch: 2 | Training loss: 0.416168 | Validation loss: 0.518829
Epoch: 3 | Training loss: 0.364288 | Validation loss: 0.496018
Epoch: 4 | Training loss: 0.326064 | Validation loss: 0.478729
Epoch: 5 | Training loss: 0.295907 | Validation loss: 0.464445
Epoch: 6 | Training loss: 0.271674 | Validation loss: 0.453058
Epoch: 7 | Training loss: 0.251322 | Validation loss: 0.443290
Epoch: 8 | Training loss: 0.234251 | Validation loss: 0.434850
Epoch: 9 | Training loss: 0.219420 | Validation loss: 0.427794
Epoch: 10 | Training loss: 0.206445 | Validation loss: 0.421439
Epoch: 11 | Training loss: 0.195075 | Validation loss: 0.415901
Epoch: 12 | Training loss: 0.184735 | Validation loss: 0.411163
Epoch: 13 | Training loss: 0.175831 | Validation loss: 0.406658
Epoch: 14 | Training loss: 0.167590 | Validation loss: 0.402873
Epoch: 15 | Training loss: 0.160046 | Validation loss: 0.399387
Epoch: 16 | Training loss: 0.153221 | Validation loss: 0.396397
Epoch: 17 | Training loss: 0.147066 | Validation loss: 0.393370
Epoch: 18 | Training loss: 0.141323 | Validation loss: 0.390855
Epoch: 19 | Training loss: 0.136002 | Validation loss: 0.388535
Epoch: 20 | Training loss: 0.131096 | Validation loss: 0.386326
Epoch: 21 | Training loss: 0.126505 | Validation loss: 0.384335
Epoch: 22 | Training loss: 0.122269 | Validation loss: 0.382568
Epoch: 23 | Training loss: 0.118276 | Validation loss: 0.380904
Epoch: 24 | Training loss: 0.114532 | Validation loss: 0.379329
Epoch: 25 | Training loss: 0.111039 | Validation loss: 0.377924
Epoch: 26 | Training loss: 0.107706 | Validation loss: 0.376605
Epoch: 27 | Training loss: 0.104634 | Validation loss: 0.375432
Epoch: 28 | Training loss: 0.101692 | Validation loss: 0.374373
Epoch: 29 | Training loss: 0.098921 | Validation loss: 0.373359
Epoch: 30 | Training loss: 0.096292 | Validation loss: 0.372366
Epoch: 31 | Training loss: 0.093803 | Validation loss: 0.371521
Epoch: 32 | Training loss: 0.091426 | Validation loss: 0.370673
Epoch: 33 | Training loss: 0.089156 | Validation loss: 0.369987
```

```
Epoch: 34 | Training loss: 0.087034 | Validation loss: 0.369267
Epoch: 35 | Training loss: 0.084975 | Validation loss: 0.368527
Epoch: 36 | Training loss: 0.083033 | Validation loss: 0.367945
Epoch: 37 | Training loss: 0.081161 | Validation loss: 0.367391
Epoch: 38 | Training loss: 0.079381 | Validation loss: 0.366891
Epoch: 39 | Training loss: 0.077667 | Validation loss: 0.366370
Epoch: 40 | Training loss: 0.076026 | Validation loss: 0.365968
Epoch: 41 | Training loss: 0.074460 | Validation loss: 0.365520
Epoch: 42 | Training loss: 0.072951 | Validation loss: 0.365144
Epoch: 43 | Training loss: 0.071504 | Validation loss: 0.364762
Epoch: 44 | Training loss: 0.070098 | Validation loss: 0.364381
Epoch: 45 | Training loss: 0.068767 | Validation loss: 0.364058
Epoch: 46 | Training loss: 0.067476 | Validation loss: 0.363774
Epoch: 47 | Training loss: 0.066228 | Validation loss: 0.363475
Epoch: 48 | Training loss: 0.065026 | Validation loss: 0.363235
Epoch: 49 | Training loss: 0.063871 | Validation loss: 0.362977
```

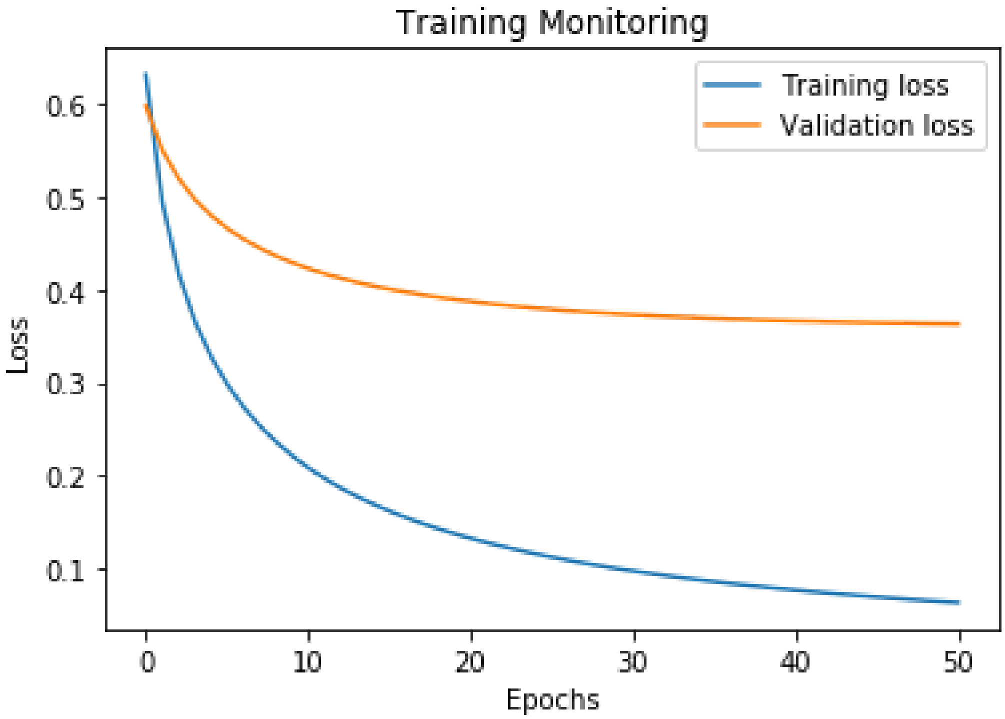
Now plot the training and validation history per epoch. Does your model underfit, overfit or is it about right? Explain why.

[34]:

```python
x = np.linspace(0,len(trl),len(trl))


plt.plot(x, trl,label='Training loss')
plt.plot(x, devl, label='Validation loss')


plt.title('Training Monitoring')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



The model have reached at the overfit situation,because the gap between the Training loss and Validation loss is very large.

Compute accuracy, precision, recall and F1-scores:

[35]:

```
# fill in your code...
preds_te = predict_class(X_test_tfidf, w_tfidf)
Y_te = data_test_label


print('Accuracy:', accuracy_score(Y_te,preds_te))
print('Precision:', precision_score(Y_te,preds_te))
print('Recall:', recall_score(Y_te,preds_te))
print('F1-Score:', f1_score(Y_te,preds_te))
```

```
Accuracy: 0.8675
Precision: 0.8516746411483254
Recall: 0.89
F1-Score: 0.8704156479217604
```

Print top-10 most positive and negative words:

```
[36]: # fill in your code...
      # Obtain a new dictionary to store the weight
      dic_weight = dict()
      for i,weight in enumerate(w_tfidf):
          dic_weight[vocab[i]] = weight

      # Obtain the positive words
      dic_resultP = dict()
      dic_resultN = dict()

      for word,weight in dic_weight.items():
          if weight >= 0:
              dic_resultP[word] = weight
          else:
              dic_resultN[word] = weight

      dic_resultP = dict(sorted(dic_resultP.items(),key=lambda weight:weight[1],␣
       ↪reverse=True)[:10])
      dic_resultN = dict(sorted(dic_resultN.items(),key=lambda weight:weight[1],␣
       ↪reverse=False)[:10])

      print("The top 10 positive words are %s respectively."% list(dic_resultP.
       ↪keys()))
      print("The top 10 negative words are %s respectively."% list(dic_resultN.
       ↪keys()))
```

```
The top 10 positive words are ['great', 'hilarious', 'fun', 'terrific',
'overall', 'perfectly', 'definitely', 'memorable', 'life', 'simple']
respectively.
The top 10 negative words are ['bad', 'worst', 'boring', 'supposed',
'unfortunately', 'waste', 'awful', 'poor', 'script', 'nothing'] respectively.
```

### 3.2.1 Discuss how did you choose model hyperparameters (e.g. learning rate and regularisation strength)? What is the relation between training epochs and learning rate? How the regularisation strength affects performance?

Enter your answer here...

[38]:

```
lr_set = [0.001,0.0005,0.0001]
alpha_set = [0.001,0.0005,0.0001]

for lr_sid in range(len(lr_set)):
    for alpha_sid in range(len(alpha_set)):
        w_tfidf, trl, devl = SGD(X_tr_tfidf, data_tr_label,
                                 X_dev=X_dev_tfidf,
                                 Y_dev=data_dev_label,
                                 lr=lr_set[lr_sid],
                                 alpha=alpha_set[alpha_sid],
                                 epochs=50)
        preds_te = predict_class(X_test_tfidf, w_tfidf)
        print("lr = %f, alpha= %f;
 ↪"%(lr_set[lr_sid],alpha_set[alpha_sid]),'Predict_Class:
 ↪%f'%(f1_score(Y_te,preds_te)))
```

```
Epoch: 0 | Training loss: 0.490848 | Validation loss: 0.422014
Epoch: 1 | Training loss: 0.179338 | Validation loss: 0.378674
Epoch: 2 | Training loss: 0.109270 | Validation loss: 0.371132
Epoch: 3 | Training loss: 0.081879 | Validation loss: 0.365232
Epoch: 4 | Training loss: 0.066032 | Validation loss: 0.363066
Epoch: 5 | Training loss: 0.055511 | Validation loss: 0.363308
lr = 0.001000, alpha= 0.001000; Predict_Class:0.861314
Epoch: 0 | Training loss: 0.492331 | Validation loss: 0.425297
Epoch: 1 | Training loss: 0.176187 | Validation loss: 0.387362
Epoch: 2 | Training loss: 0.109291 | Validation loss: 0.383238
Epoch: 3 | Training loss: 0.081937 | Validation loss: 0.368167
Epoch: 4 | Training loss: 0.065579 | Validation loss: 0.364402
Epoch: 5 | Training loss: 0.055040 | Validation loss: 0.363961
Epoch: 6 | Training loss: 0.047454 | Validation loss: 0.364355
lr = 0.001000, alpha= 0.000500; Predict_Class:0.866337
Epoch: 0 | Training loss: 0.494068 | Validation loss: 0.426704
```

```
Epoch: 1 | Training loss: 0.173315 | Validation loss: 0.390385
Epoch: 2 | Training loss: 0.109115 | Validation loss: 0.378187
Epoch: 3 | Training loss: 0.081341 | Validation loss: 0.369791
Epoch: 4 | Training loss: 0.065311 | Validation loss: 0.367865
Epoch: 5 | Training loss: 0.054376 | Validation loss: 0.365924
Epoch: 6 | Training loss: 0.046644 | Validation loss: 0.369589
lr = 0.001000, alpha= 0.000100; Predict_Class:0.864078
Epoch: 0 | Training loss: 0.530480 | Validation loss: 0.473739
Epoch: 1 | Training loss: 0.268423 | Validation loss: 0.423822
Epoch: 2 | Training loss: 0.189719 | Validation loss: 0.400644
Epoch: 3 | Training loss: 0.148375 | Validation loss: 0.391764
Epoch: 4 | Training loss: 0.122753 | Validation loss: 0.379869
Epoch: 5 | Training loss: 0.104440 | Validation loss: 0.374024
Epoch: 6 | Training loss: 0.091104 | Validation loss: 0.370380
Epoch: 7 | Training loss: 0.080774 | Validation loss: 0.367905
Epoch: 8 | Training loss: 0.072655 | Validation loss: 0.365872
Epoch: 9 | Training loss: 0.066095 | Validation loss: 0.364131
Epoch: 10 | Training loss: 0.060632 | Validation loss: 0.363386
Epoch: 11 | Training loss: 0.056028 | Validation loss: 0.363041
Epoch: 12 | Training loss: 0.052141 | Validation loss: 0.362894
Epoch: 13 | Training loss: 0.048758 | Validation loss: 0.362838
lr = 0.000500, alpha= 0.001000; Predict_Class:0.864198
Epoch: 0 | Training loss: 0.525952 | Validation loss: 0.475222
Epoch: 1 | Training loss: 0.270389 | Validation loss: 0.421016
Epoch: 2 | Training loss: 0.189989 | Validation loss: 0.397285
Epoch: 3 | Training loss: 0.148396 | Validation loss: 0.385271
Epoch: 4 | Training loss: 0.121986 | Validation loss: 0.377747
Epoch: 5 | Training loss: 0.104137 | Validation loss: 0.371402
Epoch: 6 | Training loss: 0.090657 | Validation loss: 0.367627
Epoch: 7 | Training loss: 0.080333 | Validation loss: 0.365394
Epoch: 8 | Training loss: 0.072127 | Validation loss: 0.363618
Epoch: 9 | Training loss: 0.065504 | Validation loss: 0.362583
Epoch: 10 | Training loss: 0.059988 | Validation loss: 0.361900
Epoch: 11 | Training loss: 0.055323 | Validation loss: 0.361397
Epoch: 12 | Training loss: 0.051402 | Validation loss: 0.361224
Epoch: 13 | Training loss: 0.047974 | Validation loss: 0.361317
lr = 0.000500, alpha= 0.000500; Predict_Class:0.866995
```

```
Epoch: 0 | Training loss: 0.530445 | Validation loss: 0.464269
Epoch: 1 | Training loss: 0.268302 | Validation loss: 0.416873
Epoch: 2 | Training loss: 0.189291 | Validation loss: 0.401766
Epoch: 3 | Training loss: 0.148315 | Validation loss: 0.383875
Epoch: 4 | Training loss: 0.121990 | Validation loss: 0.376471
Epoch: 5 | Training loss: 0.103667 | Validation loss: 0.370455
Epoch: 6 | Training loss: 0.090249 | Validation loss: 0.367373
Epoch: 7 | Training loss: 0.079840 | Validation loss: 0.365303
Epoch: 8 | Training loss: 0.071625 | Validation loss: 0.363942
Epoch: 9 | Training loss: 0.064853 | Validation loss: 0.364041
lr = 0.000500, alpha= 0.000100; Predict_Class:0.866180
Epoch: 0 | Training loss: 0.630648 | Validation loss: 0.597692
Epoch: 1 | Training loss: 0.492887 | Validation loss: 0.549601
Epoch: 2 | Training loss: 0.416346 | Validation loss: 0.518746
Epoch: 3 | Training loss: 0.364555 | Validation loss: 0.496064
Epoch: 4 | Training loss: 0.326253 | Validation loss: 0.478117
Epoch: 5 | Training loss: 0.296074 | Validation loss: 0.464310
Epoch: 6 | Training loss: 0.271874 | Validation loss: 0.452527
Epoch: 7 | Training loss: 0.251645 | Validation loss: 0.442786
Epoch: 8 | Training loss: 0.234474 | Validation loss: 0.434592
Epoch: 9 | Training loss: 0.219702 | Validation loss: 0.427568
Epoch: 10 | Training loss: 0.206752 | Validation loss: 0.421386
Epoch: 11 | Training loss: 0.195317 | Validation loss: 0.415953
Epoch: 12 | Training loss: 0.185297 | Validation loss: 0.410943
Epoch: 13 | Training loss: 0.176200 | Validation loss: 0.406562
Epoch: 14 | Training loss: 0.167997 | Validation loss: 0.402688
Epoch: 15 | Training loss: 0.160526 | Validation loss: 0.399364
Epoch: 16 | Training loss: 0.153758 | Validation loss: 0.396149
Epoch: 17 | Training loss: 0.147560 | Validation loss: 0.393273
Epoch: 18 | Training loss: 0.141840 | Validation loss: 0.390690
Epoch: 19 | Training loss: 0.136570 | Validation loss: 0.388328
Epoch: 20 | Training loss: 0.131660 | Validation loss: 0.386190
Epoch: 21 | Training loss: 0.127068 | Validation loss: 0.384394
Epoch: 22 | Training loss: 0.122892 | Validation loss: 0.382542
Epoch: 23 | Training loss: 0.118922 | Validation loss: 0.380867
Epoch: 24 | Training loss: 0.115206 | Validation loss: 0.379253
Epoch: 25 | Training loss: 0.111732 | Validation loss: 0.377842
```

```
Epoch: 26 | Training loss: 0.108456 | Validation loss: 0.376516
Epoch: 27 | Training loss: 0.105381 | Validation loss: 0.375320
Epoch: 28 | Training loss: 0.102460 | Validation loss: 0.374212
Epoch: 29 | Training loss: 0.099726 | Validation loss: 0.373198
Epoch: 30 | Training loss: 0.097110 | Validation loss: 0.372263
Epoch: 31 | Training loss: 0.094642 | Validation loss: 0.371367
Epoch: 32 | Training loss: 0.092285 | Validation loss: 0.370559
Epoch: 33 | Training loss: 0.090071 | Validation loss: 0.369790
Epoch: 34 | Training loss: 0.087940 | Validation loss: 0.369093
Epoch: 35 | Training loss: 0.085902 | Validation loss: 0.368481
Epoch: 36 | Training loss: 0.083992 | Validation loss: 0.367818
Epoch: 37 | Training loss: 0.082147 | Validation loss: 0.367236
Epoch: 38 | Training loss: 0.080383 | Validation loss: 0.366715
Epoch: 39 | Training loss: 0.078691 | Validation loss: 0.366188
Epoch: 40 | Training loss: 0.077071 | Validation loss: 0.365700
Epoch: 41 | Training loss: 0.075524 | Validation loss: 0.365278
Epoch: 42 | Training loss: 0.074032 | Validation loss: 0.364920
Epoch: 43 | Training loss: 0.072605 | Validation loss: 0.364529
Epoch: 44 | Training loss: 0.071217 | Validation loss: 0.364192
Epoch: 45 | Training loss: 0.069910 | Validation loss: 0.363865
Epoch: 46 | Training loss: 0.068645 | Validation loss: 0.363564
Epoch: 47 | Training loss: 0.067389 | Validation loss: 0.363356
Epoch: 48 | Training loss: 0.066227 | Validation loss: 0.363043
Epoch: 49 | Training loss: 0.065097 | Validation loss: 0.362808
lr = 0.000100, alpha= 0.001000; Predict_Class:0.869779
Epoch: 0 | Training loss: 0.629683 | Validation loss: 0.598444
Epoch: 1 | Training loss: 0.492813 | Validation loss: 0.549464
Epoch: 2 | Training loss: 0.416894 | Validation loss: 0.518050
Epoch: 3 | Training loss: 0.364683 | Validation loss: 0.496071
Epoch: 4 | Training loss: 0.325984 | Validation loss: 0.478819
Epoch: 5 | Training loss: 0.296142 | Validation loss: 0.464652
Epoch: 6 | Training loss: 0.271887 | Validation loss: 0.452846
Epoch: 7 | Training loss: 0.251594 | Validation loss: 0.443159
Epoch: 8 | Training loss: 0.234339 | Validation loss: 0.434699
Epoch: 9 | Training loss: 0.219605 | Validation loss: 0.427679
Epoch: 10 | Training loss: 0.206682 | Validation loss: 0.421415
Epoch: 11 | Training loss: 0.195259 | Validation loss: 0.415930
```

```
Epoch: 12 | Training loss: 0.185059 | Validation loss: 0.411120
Epoch: 13 | Training loss: 0.176049 | Validation loss: 0.406791
Epoch: 14 | Training loss: 0.167775 | Validation loss: 0.402894
Epoch: 15 | Training loss: 0.160169 | Validation loss: 0.399585
Epoch: 16 | Training loss: 0.153589 | Validation loss: 0.396217
Epoch: 17 | Training loss: 0.147341 | Validation loss: 0.393365
Epoch: 18 | Training loss: 0.141567 | Validation loss: 0.390807
Epoch: 19 | Training loss: 0.136311 | Validation loss: 0.388407
Epoch: 20 | Training loss: 0.131382 | Validation loss: 0.386209
Epoch: 21 | Training loss: 0.126832 | Validation loss: 0.384290
Epoch: 22 | Training loss: 0.122541 | Validation loss: 0.382424
Epoch: 23 | Training loss: 0.118607 | Validation loss: 0.380771
Epoch: 24 | Training loss: 0.114888 | Validation loss: 0.379330
Epoch: 25 | Training loss: 0.111400 | Validation loss: 0.377933
Epoch: 26 | Training loss: 0.108108 | Validation loss: 0.376662
Epoch: 27 | Training loss: 0.105007 | Validation loss: 0.375498
Epoch: 28 | Training loss: 0.102087 | Validation loss: 0.374284
Epoch: 29 | Training loss: 0.099319 | Validation loss: 0.373285
Epoch: 30 | Training loss: 0.096713 | Validation loss: 0.372282
Epoch: 31 | Training loss: 0.094228 | Validation loss: 0.371378
Epoch: 32 | Training loss: 0.091871 | Validation loss: 0.370556
Epoch: 33 | Training loss: 0.089624 | Validation loss: 0.369776
Epoch: 34 | Training loss: 0.087486 | Validation loss: 0.369059
Epoch: 35 | Training loss: 0.085456 | Validation loss: 0.368410
Epoch: 36 | Training loss: 0.083508 | Validation loss: 0.367790
Epoch: 37 | Training loss: 0.081658 | Validation loss: 0.367220
Epoch: 38 | Training loss: 0.079853 | Validation loss: 0.366734
Epoch: 39 | Training loss: 0.078182 | Validation loss: 0.366165
Epoch: 40 | Training loss: 0.076554 | Validation loss: 0.365695
Epoch: 41 | Training loss: 0.074992 | Validation loss: 0.365289
Epoch: 42 | Training loss: 0.073493 | Validation loss: 0.364882
Epoch: 43 | Training loss: 0.072051 | Validation loss: 0.364500
Epoch: 44 | Training loss: 0.070670 | Validation loss: 0.364161
Epoch: 45 | Training loss: 0.069328 | Validation loss: 0.363809
Epoch: 46 | Training loss: 0.068054 | Validation loss: 0.363505
Epoch: 47 | Training loss: 0.066814 | Validation loss: 0.363223
Epoch: 48 | Training loss: 0.065618 | Validation loss: 0.363024
```

```
Epoch: 49 | Training loss: 0.064479 | Validation loss: 0.362802
lr = 0.000100, alpha= 0.000500; Predict_Class:0.868293
Epoch: 0 | Training loss: 0.631021 | Validation loss: 0.598696
Epoch: 1 | Training loss: 0.493515 | Validation loss: 0.549354
Epoch: 2 | Training loss: 0.416807 | Validation loss: 0.517926
Epoch: 3 | Training loss: 0.364625 | Validation loss: 0.495753
Epoch: 4 | Training loss: 0.326144 | Validation loss: 0.478580
Epoch: 5 | Training loss: 0.296179 | Validation loss: 0.464246
Epoch: 6 | Training loss: 0.271745 | Validation loss: 0.452898
Epoch: 7 | Training loss: 0.251291 | Validation loss: 0.443203
Epoch: 8 | Training loss: 0.234339 | Validation loss: 0.434737
Epoch: 9 | Training loss: 0.219472 | Validation loss: 0.427603
Epoch: 10 | Training loss: 0.206497 | Validation loss: 0.421511
Epoch: 11 | Training loss: 0.195106 | Validation loss: 0.416031
Epoch: 12 | Training loss: 0.184951 | Validation loss: 0.411158
Epoch: 13 | Training loss: 0.175868 | Validation loss: 0.406794
Epoch: 14 | Training loss: 0.167600 | Validation loss: 0.402912
Epoch: 15 | Training loss: 0.160141 | Validation loss: 0.399466
Epoch: 16 | Training loss: 0.153337 | Validation loss: 0.396372
Epoch: 17 | Training loss: 0.147122 | Validation loss: 0.393490
Epoch: 18 | Training loss: 0.141353 | Validation loss: 0.390911
Epoch: 19 | Training loss: 0.136057 | Validation loss: 0.388517
Epoch: 20 | Training loss: 0.131130 | Validation loss: 0.386454
Epoch: 21 | Training loss: 0.126575 | Validation loss: 0.384496
Epoch: 22 | Training loss: 0.122295 | Validation loss: 0.382750
Epoch: 23 | Training loss: 0.118314 | Validation loss: 0.381120
Epoch: 24 | Training loss: 0.114600 | Validation loss: 0.379520
Epoch: 25 | Training loss: 0.111085 | Validation loss: 0.378076
Epoch: 26 | Training loss: 0.107765 | Validation loss: 0.376791
Epoch: 27 | Training loss: 0.104701 | Validation loss: 0.375590
Epoch: 28 | Training loss: 0.101762 | Validation loss: 0.374484
Epoch: 29 | Training loss: 0.098995 | Validation loss: 0.373433
Epoch: 30 | Training loss: 0.096369 | Validation loss: 0.372402
Epoch: 31 | Training loss: 0.093876 | Validation loss: 0.371507
Epoch: 32 | Training loss: 0.091479 | Validation loss: 0.370758
Epoch: 33 | Training loss: 0.089257 | Validation loss: 0.369907
Epoch: 34 | Training loss: 0.087105 | Validation loss: 0.369222
```

```
Epoch: 35 | Training loss: 0.085068 | Validation loss: 0.368578
Epoch: 36 | Training loss: 0.083113 | Validation loss: 0.367987
Epoch: 37 | Training loss: 0.081254 | Validation loss: 0.367411
Epoch: 38 | Training loss: 0.079468 | Validation loss: 0.366887
Epoch: 39 | Training loss: 0.077760 | Validation loss: 0.366374
Epoch: 40 | Training loss: 0.076124 | Validation loss: 0.365914
Epoch: 41 | Training loss: 0.074544 | Validation loss: 0.365460
Epoch: 42 | Training loss: 0.073049 | Validation loss: 0.365051
Epoch: 43 | Training loss: 0.071593 | Validation loss: 0.364705
Epoch: 44 | Training loss: 0.070206 | Validation loss: 0.364347
Epoch: 45 | Training loss: 0.068860 | Validation loss: 0.364000
Epoch: 46 | Training loss: 0.067574 | Validation loss: 0.363682
Epoch: 47 | Training loss: 0.066335 | Validation loss: 0.363417
Epoch: 48 | Training loss: 0.065132 | Validation loss: 0.363194
Epoch: 49 | Training loss: 0.063978 | Validation loss: 0.362955
lr = 0.000100, alpha= 0.000100; Predict_Class:0.870416
```

From the previous result, we can find that whtn the lr=0.0001,alpha=0.0001, the value of the F1-Score has reached at the peak.

So we conclude that the lr has more influence on the model training.

If we want to avoid the overfitting, we should apply the regularization.

### 3.3 Full Results

Add here your results:

| LR | Precision | Recall | F1-Score |
|---|---|---|---|
| BOW-count | 0.825 | 0.85 | 0.83744 |
| BOW-tfidf | 0.85167 | 0.89 | 0.8704156 |

## 4 Multi-class Logistic Regression

Now you need to train a Multiclass Logistic Regression (MLR) Classifier by extending the Binary model you developed above. You will use the MLR model to perform topic classification on the AG news dataset consisting of three classes:

- Class 1: World
- Class 2: Sports
- Class 3: Business

You need to follow the same process as in Task 1 for data processing and feature extraction by reusing the functions you wrote.

[39]:
```
# fill in your code...
data_tr = pd.read_csv("./data_topic/train.csv",header=None,␣
 ↪names=['label','text'])
data_dev = pd.read_csv("./data_topic/dev.csv",header=None,␣
 ↪names=['label','text'])
data_test = pd.read_csv("./data_topic/test.csv",header=None,␣
 ↪names=['label','text'])
```

[40]:
```
data_tr.head()
```

[40]:
```
   label                                               text
0      1  Reuters - Venezuelans turned out early\and in …
1      1  Reuters - South Korean police used water canno…
2      1  Reuters - Thousands of Palestinian\prisoners i…
3      1  AFP - Sporadic gunfire and shelling took place…
4      1  AP - Dozens of Rwandan soldiers flew into Suda…
```

[41]:
```
# We can obtain the all the file into
data_tr_list = [word.lower() for word in data_tr['text'].tolist()]
data_tr_label = np.array(data_tr['label'])
# development dataset
data_dev_list = [word.lower() for word in data_dev['text'].tolist()]
data_dev_label = np.array(data_dev['label'])
# test dataset
data_test_list = [word.lower() for word in data_test['text'].tolist()]
data_test_label = np.array(data_test['label'])
```

[42]:
```
stop_words = ['a','in','on','at','and','or',
              'to', 'the', 'of', 'an', 'by',
              'as', 'is', 'was', 'were', 'been', 'be',
              'are','for', 'this', 'that', 'these', 'those', 'you', 'i',
             'it', 'he', 'she', 'we', 'they' 'will', 'have', 'has',
              'do', 'did', 'can', 'could', 'who', 'which', 'what',
             'his', 'her', 'they', 'them', 'from', 'with', 'its']
```

[43]:

```
tr_ngram = extract_ngrams(data_tr_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
dev_ngram = extract_ngrams(data_dev_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
test_ngram = extract_ngrams(data_test_list, ngram_range=(1,3),␣
 ↪stop_words=stop_words)
```

[44]:

```
vocab, df, ngram_counts = get_vocab(data_tr_list, ngram_range=(1,3),␣
 ↪keep_topN=5000, stop_words=stop_words)
print(len(vocab))
print()
print(list(vocab)[:100])
print()
print(list(df.items())[:10])
# print(df.most_common()[:10])
```

```
5000

['reuters', 'said', 'tuesday', 'wednesday', 'new', 'after', 'ap', 'athens',
'monday', 'first', 'two', 'york', 'over', ('new', 'york'), 'us', 'olympic',
'but', 'their', 'will', 'inc', 'more', 'year', 'oil', 'prices', 'company',
'world', 'than', 'aug', 'about', 'had', 'united', 'one', 'sunday', 'out',
'into', 'against', 'up', 'second', 'last', 'president', 'stocks', 'gold',
'team', ('york', 'reuters'), ('new', 'york', 'reuters'), 'when', 'three',
'night', 'time', 'no', 'yesterday', 'games', 'olympics', 'not', 'states',
'greece', 'off', 'iraq', 'washington', 'percent', ('united', 'states'), ('oil',
'prices'), 'home', 'day', 'google', 'public', ('athens', 'reuters'), 'record',
'week', 'men', 'government', 'win', ('said', 'tuesday'), 'american', 'won',
'years', 'all', 'billion', 'shares', 'city', 'offering', 'officials', 'would',
'today', 'final', 'afp', 'gt', 'people', 'lt', 'medal', 'corp', 'sales',
'country', 'back', 'four', 'high', 'investor', 'com', 'minister', 'reported']

[('reuters', 631), ('said', 432), ('tuesday', 413), ('wednesday', 344), ('new',
325), ('after', 295), ('ap', 275), ('athens', 245), ('monday', 221), ('first',
210)]
```

[45]:

```
# Vectorise
X_tr_count = vectorise(tr_ngram,vocab)
X_dev_count = vectorise(dev_ngram,vocab)
X_test_count = vectorise(test_ngram,vocab)

#Show
X_tr_count[:2,:50]
```

[45]:

```
array([[1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
        0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0.]])
```

Tf.idf Part

[46]:

```
X_tr_tfidf = tfidf_function(X_count=X_tr_count, ngram_list = tr_ngram, vocab =␣
 ↪vocab)
X_dev_tfidf = tfidf_function(X_count=X_dev_count, ngram_list = dev_ngram, vocab␣
 ↪= vocab)
X_test_tfidf = tfidf_function(X_count=X_test_count, ngram_list = test_ngram,␣
 ↪vocab = vocab)
```

Now you need to change `SGD` to support multiclass datasets. First you need to develop a `softmax` function. It takes as input:

- `z`: array of real numbers

and returns:

- `smax`: the softmax of `z`

[47]:

```
def softmax(z):

    if len(z.shape) >1:
        x_sum = np.sum(np.exp(z), axis = 1, keepdims = True)
    else:
        x_sum = np.sum(np.exp(z), axis = 0)

    smax = np.exp(z) / x_sum
    return smax
```

Then modify `predict_proba` and `predict_class` functions for the multiclass case:

[48]:

```
def predict_proba(X, weights):

    P_value = np.dot(X,weights.T)
    preds_proba = softmax(P_value)

    return preds_proba
```

[49]:

```
def predict_class(X, weights):

    pro_value = predict_proba(X,weights)
    if len(pro_value.shape)>1:
        feature_list = list()
        for value in pro_value:
            value[value ==  max(value)] = 1
            for i,row in enumerate(value):
                if row == 1:
                    feature_list.append(i+1)
                    break;
        feature_list = np.asarray(feature_list)

    else:
        pro_value[pro_value==max(pro_value)] = 1
        for i,value in enumerate(pro_value):
            if value == 1:
                feature_list = i+1
                break;

    return feature_list
```

Toy example and expected functionality of the functions above:

[50]:
```
X = np.array([[0.1,0.2],[0.2,0.1],[0.1,-0.2]])
w = np.array([[2,-5],[-5,2]])
```

[51]:
```
predict_proba(X, w)
```

[51]:
```
array([[0.33181223, 0.66818777],
       [0.66818777, 0.33181223],
       [0.89090318, 0.10909682]])
```

[52]:
```
predict_class(X, w)
```

[52]:
```
array([2, 1, 1])
```

Now you need to compute the categorical cross entropy loss (extending the binary loss to support multiple classes).

[53]:
```
def categorical_loss(X, Y, weights, num_classes=5, alpha=0.00001):

    y_proba = predict_proba(X,weights)
    # Use the np.clip to restrict the value range
    y_proba = np.clip(y_proba,alpha,1-alpha)

    if len(X.shape)>1:
        loss_value = -np.log(y_proba[range(len(Y)),Y-1])
        L2_reg_value = (1/len(X))*(alpha/2)*(np.sum(np.square(weights)))
    else:
        loss_value = -np.log(y_proba[Y-1])
        L2_reg_value = (alpha/2)*(np.sum(np.square(weights)))

    l = loss_value + L2_reg_value
    return l
```

[54]:

```
def one_regularize(X,class_nums):


    return np.delete(np.eye(class_nums+1)[X],0,axis=1)
```

Finally you need to modify SGD to support the categorical cross entropy loss:

[55]:

```
def SGD(X_tr, Y_tr, vocab, X_dev=[], Y_dev=[], num_classes=5, lr=0.01, alpha=0.
↪00001, epochs=5, tolerance=0.001, print_progress=True):


    cur_loss_dev = 2.
    training_loss_history = []
    validation_loss_history = []

    # Obtain the weights
    weights = np.zeros((3, len(vocab)))
    Y_tr_onehot = one_regularize(Y_tr,num_classes)

    for i in range(epochs):
        loss_list = list()
        seed_number = random.randint(50,100)
        np.random.seed(seed_number)
        # Shuffle the X_tr and Y_tr
        per_X_tr = np.random.permutation(X_tr)
        np.random.seed(seed_number)
        per_Y_tr = np.random.permutation(Y_tr)
        np.random.seed(seed_number)
        per_Y_tr_onehot = np.random.permutation(Y_tr_onehot)

        for j,row in enumerate(per_X_tr):
            # Caculate the categorical_loss and store the loss into loss_list
            loss_tr = categorical_loss(row, per_Y_tr[j], weights, alpha)
            loss_list.append(loss_tr)
            y_pred = predict_proba(row,weights)
            error = y_pred - per_Y_tr_onehot[j]
            # update weights
            weights_list = [weights[i]-lr*row*error[i] for i in␣
↪range(num_classes)]
```

```
            weights = np.asarray(weights_list)

        # Obtain the mean
        tr_loss_mean = np.mean(loss_list)
        training_loss_history.append(tr_loss_mean)

        dev_loss = categorical_loss(X_dev, Y_dev, weights, alpha)
        dev_loss = sum(dev_loss)/len(dev_loss)
        validation_loss_history.append(dev_loss)
        print('Epoch: %d' % i, '| Training loss: %f' %tr_loss_mean, '| Validation loss: %f' %dev_loss)

        if (cur_loss_dev-dev_loss)<tolerance:
            break
        cur_loss_dev = dev_loss


    return weights, training_loss_history, validation_loss_history
```

Now you are ready to train and evaluate you MLR following the same steps as in Task 1 for both Count and tfidf features:

[56]:

```
w_count, loss_tr_count, dev_loss_count = SGD(X_tr_count, data_tr_label,vocab,
                                             X_dev=X_dev_count,
                                             Y_dev=data_dev_label,
                                             num_classes=3,
                                             lr=0.0001,
                                             alpha=0.001,
                                             epochs=200)
```

```
Epoch: 0 | Training loss: 1.084229 | Validation loss: 1.083282
Epoch: 1 | Training loss: 1.056024 | Validation loss: 1.069042
Epoch: 2 | Training loss: 1.030918 | Validation loss: 1.055699
Epoch: 3 | Training loss: 1.008251 | Validation loss: 1.043099
Epoch: 4 | Training loss: 0.987531 | Validation loss: 1.031134
Epoch: 5 | Training loss: 0.968390 | Validation loss: 1.019712
Epoch: 6 | Training loss: 0.950563 | Validation loss: 1.008770
```

```
Epoch: 7 | Training loss: 0.933864 | Validation loss: 0.998258
Epoch: 8 | Training loss: 0.918148 | Validation loss: 0.988138
Epoch: 9 | Training loss: 0.903305 | Validation loss: 0.978378
Epoch: 10 | Training loss: 0.889241 | Validation loss: 0.968950
Epoch: 11 | Training loss: 0.875882 | Validation loss: 0.959835
Epoch: 12 | Training loss: 0.863166 | Validation loss: 0.951012
Epoch: 13 | Training loss: 0.851038 | Validation loss: 0.942466
Epoch: 14 | Training loss: 0.839455 | Validation loss: 0.934182
Epoch: 15 | Training loss: 0.828375 | Validation loss: 0.926146
Epoch: 16 | Training loss: 0.817762 | Validation loss: 0.918347
Epoch: 17 | Training loss: 0.807583 | Validation loss: 0.910774
Epoch: 18 | Training loss: 0.797808 | Validation loss: 0.903415
Epoch: 19 | Training loss: 0.788410 | Validation loss: 0.896262
Epoch: 20 | Training loss: 0.779368 | Validation loss: 0.889305
Epoch: 21 | Training loss: 0.770658 | Validation loss: 0.882538
Epoch: 22 | Training loss: 0.762262 | Validation loss: 0.875949
Epoch: 23 | Training loss: 0.754161 | Validation loss: 0.869534
Epoch: 24 | Training loss: 0.746338 | Validation loss: 0.863282
Epoch: 25 | Training loss: 0.738776 | Validation loss: 0.857191
Epoch: 26 | Training loss: 0.731461 | Validation loss: 0.851252
Epoch: 27 | Training loss: 0.724379 | Validation loss: 0.845460
Epoch: 28 | Training loss: 0.717520 | Validation loss: 0.839810
Epoch: 29 | Training loss: 0.710871 | Validation loss: 0.834295
Epoch: 30 | Training loss: 0.704422 | Validation loss: 0.828912
Epoch: 31 | Training loss: 0.698162 | Validation loss: 0.823653
Epoch: 32 | Training loss: 0.692083 | Validation loss: 0.818516
Epoch: 33 | Training loss: 0.686175 | Validation loss: 0.813497
Epoch: 34 | Training loss: 0.680432 | Validation loss: 0.808592
Epoch: 35 | Training loss: 0.674846 | Validation loss: 0.803794
Epoch: 36 | Training loss: 0.669408 | Validation loss: 0.799102
Epoch: 37 | Training loss: 0.664112 | Validation loss: 0.794512
Epoch: 38 | Training loss: 0.658952 | Validation loss: 0.790020
Epoch: 39 | Training loss: 0.653923 | Validation loss: 0.785622
Epoch: 40 | Training loss: 0.649019 | Validation loss: 0.781318
Epoch: 41 | Training loss: 0.644235 | Validation loss: 0.777101
Epoch: 42 | Training loss: 0.639565 | Validation loss: 0.772970
Epoch: 43 | Training loss: 0.635005 | Validation loss: 0.768922
```

```
Epoch: 44 | Training loss: 0.630551 | Validation loss: 0.764955
Epoch: 45 | Training loss: 0.626198 | Validation loss: 0.761065
Epoch: 46 | Training loss: 0.621943 | Validation loss: 0.757252
Epoch: 47 | Training loss: 0.617783 | Validation loss: 0.753511
Epoch: 48 | Training loss: 0.613712 | Validation loss: 0.749842
Epoch: 49 | Training loss: 0.609729 | Validation loss: 0.746240
Epoch: 50 | Training loss: 0.605829 | Validation loss: 0.742705
Epoch: 51 | Training loss: 0.602010 | Validation loss: 0.739235
Epoch: 52 | Training loss: 0.598270 | Validation loss: 0.735828
Epoch: 53 | Training loss: 0.594605 | Validation loss: 0.732482
Epoch: 54 | Training loss: 0.591012 | Validation loss: 0.729195
Epoch: 55 | Training loss: 0.587489 | Validation loss: 0.725966
Epoch: 56 | Training loss: 0.584035 | Validation loss: 0.722793
Epoch: 57 | Training loss: 0.580646 | Validation loss: 0.719675
Epoch: 58 | Training loss: 0.577320 | Validation loss: 0.716608
Epoch: 59 | Training loss: 0.574056 | Validation loss: 0.713594
Epoch: 60 | Training loss: 0.570851 | Validation loss: 0.710630
Epoch: 61 | Training loss: 0.567705 | Validation loss: 0.707714
Epoch: 62 | Training loss: 0.564613 | Validation loss: 0.704846
Epoch: 63 | Training loss: 0.561577 | Validation loss: 0.702024
Epoch: 64 | Training loss: 0.558592 | Validation loss: 0.699248
Epoch: 65 | Training loss: 0.555659 | Validation loss: 0.696515
Epoch: 66 | Training loss: 0.552775 | Validation loss: 0.693825
Epoch: 67 | Training loss: 0.549939 | Validation loss: 0.691177
Epoch: 68 | Training loss: 0.547150 | Validation loss: 0.688569
Epoch: 69 | Training loss: 0.544405 | Validation loss: 0.686001
Epoch: 70 | Training loss: 0.541706 | Validation loss: 0.683472
Epoch: 71 | Training loss: 0.539049 | Validation loss: 0.680980
Epoch: 72 | Training loss: 0.536433 | Validation loss: 0.678524
Epoch: 73 | Training loss: 0.533858 | Validation loss: 0.676105
Epoch: 74 | Training loss: 0.531322 | Validation loss: 0.673721
Epoch: 75 | Training loss: 0.528825 | Validation loss: 0.671372
Epoch: 76 | Training loss: 0.526365 | Validation loss: 0.669054
Epoch: 77 | Training loss: 0.523941 | Validation loss: 0.666770
Epoch: 78 | Training loss: 0.521553 | Validation loss: 0.664518
Epoch: 79 | Training loss: 0.519199 | Validation loss: 0.662297
Epoch: 80 | Training loss: 0.516879 | Validation loss: 0.660107
```

```
Epoch: 81 | Training loss: 0.514592 | Validation loss: 0.657946
Epoch: 82 | Training loss: 0.512337 | Validation loss: 0.655814
Epoch: 83 | Training loss: 0.510113 | Validation loss: 0.653712
Epoch: 84 | Training loss: 0.507920 | Validation loss: 0.651636
Epoch: 85 | Training loss: 0.505756 | Validation loss: 0.649589
Epoch: 86 | Training loss: 0.503622 | Validation loss: 0.647569
Epoch: 87 | Training loss: 0.501516 | Validation loss: 0.645573
Epoch: 88 | Training loss: 0.499437 | Validation loss: 0.643603
Epoch: 89 | Training loss: 0.497386 | Validation loss: 0.641658
Epoch: 90 | Training loss: 0.495361 | Validation loss: 0.639739
Epoch: 91 | Training loss: 0.493362 | Validation loss: 0.637843
Epoch: 92 | Training loss: 0.491388 | Validation loss: 0.635971
Epoch: 93 | Training loss: 0.489439 | Validation loss: 0.634122
Epoch: 94 | Training loss: 0.487514 | Validation loss: 0.632296
Epoch: 95 | Training loss: 0.485613 | Validation loss: 0.630491
Epoch: 96 | Training loss: 0.483735 | Validation loss: 0.628708
Epoch: 97 | Training loss: 0.481880 | Validation loss: 0.626946
Epoch: 98 | Training loss: 0.480047 | Validation loss: 0.625206
Epoch: 99 | Training loss: 0.478235 | Validation loss: 0.623486
Epoch: 100 | Training loss: 0.476445 | Validation loss: 0.621786
Epoch: 101 | Training loss: 0.474676 | Validation loss: 0.620106
Epoch: 102 | Training loss: 0.472927 | Validation loss: 0.618445
Epoch: 103 | Training loss: 0.471198 | Validation loss: 0.616803
Epoch: 104 | Training loss: 0.469489 | Validation loss: 0.615180
Epoch: 105 | Training loss: 0.467799 | Validation loss: 0.613575
Epoch: 106 | Training loss: 0.466128 | Validation loss: 0.611987
Epoch: 107 | Training loss: 0.464475 | Validation loss: 0.610417
Epoch: 108 | Training loss: 0.462840 | Validation loss: 0.608866
Epoch: 109 | Training loss: 0.461223 | Validation loss: 0.607330
Epoch: 110 | Training loss: 0.459624 | Validation loss: 0.605810
Epoch: 111 | Training loss: 0.458041 | Validation loss: 0.604308
Epoch: 112 | Training loss: 0.456476 | Validation loss: 0.602821
Epoch: 113 | Training loss: 0.454927 | Validation loss: 0.601352
Epoch: 114 | Training loss: 0.453394 | Validation loss: 0.599897
Epoch: 115 | Training loss: 0.451877 | Validation loss: 0.598457
Epoch: 116 | Training loss: 0.450375 | Validation loss: 0.597033
Epoch: 117 | Training loss: 0.448889 | Validation loss: 0.595623
```

```
Epoch: 118 | Training loss: 0.447417 | Validation loss: 0.594228
Epoch: 119 | Training loss: 0.445961 | Validation loss: 0.592847
Epoch: 120 | Training loss: 0.444519 | Validation loss: 0.591480
Epoch: 121 | Training loss: 0.443091 | Validation loss: 0.590127
Epoch: 122 | Training loss: 0.441677 | Validation loss: 0.588787
Epoch: 123 | Training loss: 0.440277 | Validation loss: 0.587460
Epoch: 124 | Training loss: 0.438891 | Validation loss: 0.586147
Epoch: 125 | Training loss: 0.437518 | Validation loss: 0.584847
Epoch: 126 | Training loss: 0.436158 | Validation loss: 0.583559
Epoch: 127 | Training loss: 0.434810 | Validation loss: 0.582284
Epoch: 128 | Training loss: 0.433476 | Validation loss: 0.581021
Epoch: 129 | Training loss: 0.432154 | Validation loss: 0.579771
Epoch: 130 | Training loss: 0.430844 | Validation loss: 0.578532
Epoch: 131 | Training loss: 0.429546 | Validation loss: 0.577305
Epoch: 132 | Training loss: 0.428260 | Validation loss: 0.576089
Epoch: 133 | Training loss: 0.426985 | Validation loss: 0.574884
Epoch: 134 | Training loss: 0.425722 | Validation loss: 0.573691
Epoch: 135 | Training loss: 0.424471 | Validation loss: 0.572509
Epoch: 136 | Training loss: 0.423230 | Validation loss: 0.571338
Epoch: 137 | Training loss: 0.422000 | Validation loss: 0.570178
Epoch: 138 | Training loss: 0.420782 | Validation loss: 0.569028
Epoch: 139 | Training loss: 0.419573 | Validation loss: 0.567889
Epoch: 140 | Training loss: 0.418376 | Validation loss: 0.566760
Epoch: 141 | Training loss: 0.417188 | Validation loss: 0.565641
Epoch: 142 | Training loss: 0.416011 | Validation loss: 0.564532
Epoch: 143 | Training loss: 0.414843 | Validation loss: 0.563431
Epoch: 144 | Training loss: 0.413686 | Validation loss: 0.562342
Epoch: 145 | Training loss: 0.412538 | Validation loss: 0.561261
Epoch: 146 | Training loss: 0.411399 | Validation loss: 0.560190
Epoch: 147 | Training loss: 0.410270 | Validation loss: 0.559128
Epoch: 148 | Training loss: 0.409151 | Validation loss: 0.558075
Epoch: 149 | Training loss: 0.408040 | Validation loss: 0.557031
Epoch: 150 | Training loss: 0.406939 | Validation loss: 0.555996
Epoch: 151 | Training loss: 0.405846 | Validation loss: 0.554970
Epoch: 152 | Training loss: 0.404762 | Validation loss: 0.553953
Epoch: 153 | Training loss: 0.403687 | Validation loss: 0.552944
Epoch: 154 | Training loss: 0.402620 | Validation loss: 0.551943
```

```
Epoch: 155 | Training loss: 0.401562 | Validation loss: 0.550951
```

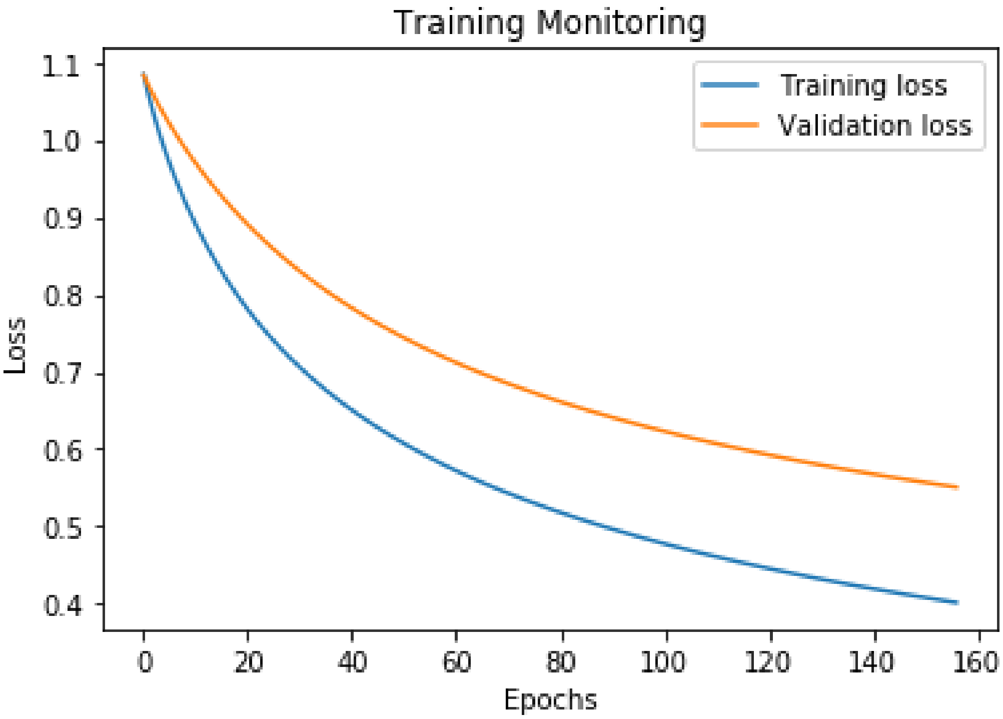
Plot training and validation process and explain if your model overfit, underfit or is about right:

[57]:

```
x = np.linspace(0,len(loss_tr_count),len(loss_tr_count))
y1, y2 = loss_tr_count, dev_loss_count

plt.plot(x, y1,label='Training loss')
plt.plot(x, y2, label='Validation loss')

plt.title('Training Monitoring')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



The model is overfit

Compute accuracy, precision, recall and F1-scores:

[58]:

```
# fill in your code...
preds_te_count = predict_class(X = X_test_count, weights = w_count)
Y_te = data_test_label

print('Accuracy:', accuracy_score(Y_te,preds_te_count))
print('Precision:', precision_score(Y_te,preds_te_count))
print('Recall:', recall_score(Y_te,preds_te_count))
print('F1-Score:', f1_score(Y_te,preds_te_count))
```

```
Accuracy: 0.8533333333333334
Precision: 0.8233333333333334
Recall: 0.8233333333333334
F1-Score: 0.8233333333333334
```

Print the top-10 words for each class respectively.

[59]:

```
for i in range(len(w_count)):

    dict_store = {vocab[sid]:weight for sid,weight in enumerate(w_count[i])}
    # Order By
    dict_store = dict(sorted(dict_store.items(), key=lambda col:
 ↪col[1],reverse=True)[:10])

    print("The top 10 words for class %d are %s respectively.
 ↪"%(i+1,list(dict_store.keys())))
```

```
The top 10 words for class 1 are ['afp', 'said', 'president', 'minister',
'najaf', 'people', 'monday', 'iraq', 'troops', 'al'] respectively.
The top 10 words for class 2 are ['athens', 'olympic', 'ap', 'team', ('athens',
'reuters'), 'win', 'games', 'olympics', 'game', 'when'] respectively.
The top 10 words for class 3 are ['company', 'inc', 'oil', 'corp', 'billion',
'prices', 'business', 'sales', 'million', 'market'] respectively.
```

### 4.0.1 Discuss how did you choose model hyperparameters (e.g. learning rate and regularisation strength)? What is the relation between training epochs and learning rate? How the regularisation strength affects performance?

Explain here...

[62]:

```
lr_set = [0.001,0.0005,0.0001]
alpha_set = [0.001,0.0005,0.0001]

for lr_sid in range(len(lr_set)):
    for alpha_sid in range(len(alpha_set)):
        w_count, loss_tr_count, dev_loss_count = SGD(X_tr_count,␣
 ↪data_tr_label,vocab,
                                                          X_dev=X_dev_count,
                                                          Y_dev=data_dev_label,
                                                          num_classes=3,
                                                          lr=lr_set[lr_sid],
                                                          alpha=alpha_set[alpha_sid],
                                                          epochs=200)
        preds_te = predict_class(X = X_test_count, weights = w_count)
        print("lr = %f, alpha= %f;
 ↪"%(lr_set[lr_sid],alpha_set[alpha_sid]),'Predict_Class:
 ↪%f'%(f1_score(Y_te,preds_te)))
```

```
Epoch: 0 | Training loss: 0.990362 | Validation loss: 0.977769
Epoch: 1 | Training loss: 0.840230 | Validation loss: 0.895535
Epoch: 2 | Training loss: 0.747093 | Validation loss: 0.833415
Epoch: 3 | Training loss: 0.681181 | Validation loss: 0.784788
Epoch: 4 | Training loss: 0.631315 | Validation loss: 0.745370
Epoch: 5 | Training loss: 0.591804 | Validation loss: 0.712715
Epoch: 6 | Training loss: 0.559436 | Validation loss: 0.685196
Epoch: 7 | Training loss: 0.532167 | Validation loss: 0.661514
Epoch: 8 | Training loss: 0.508784 | Validation loss: 0.640886
Epoch: 9 | Training loss: 0.488371 | Validation loss: 0.622766
Epoch: 10 | Training loss: 0.470339 | Validation loss: 0.606647
Epoch: 11 | Training loss: 0.454241 | Validation loss: 0.592179
Epoch: 12 | Training loss: 0.439729 | Validation loss: 0.579122
Epoch: 13 | Training loss: 0.426550 | Validation loss: 0.567233
```

```
Epoch: 14 | Training loss: 0.414496 | Validation loss: 0.556338
Epoch: 15 | Training loss: 0.403425 | Validation loss: 0.546391
Epoch: 16 | Training loss: 0.393194 | Validation loss: 0.537241
Epoch: 17 | Training loss: 0.383698 | Validation loss: 0.528788
Epoch: 18 | Training loss: 0.374844 | Validation loss: 0.520894
Epoch: 19 | Training loss: 0.366565 | Validation loss: 0.513538
Epoch: 20 | Training loss: 0.358788 | Validation loss: 0.506665
Epoch: 21 | Training loss: 0.351492 | Validation loss: 0.500204
Epoch: 22 | Training loss: 0.344595 | Validation loss: 0.494105
Epoch: 23 | Training loss: 0.338077 | Validation loss: 0.488391
Epoch: 24 | Training loss: 0.331905 | Validation loss: 0.482995
Epoch: 25 | Training loss: 0.326036 | Validation loss: 0.477865
Epoch: 26 | Training loss: 0.320453 | Validation loss: 0.473042
Epoch: 27 | Training loss: 0.315135 | Validation loss: 0.468453
Epoch: 28 | Training loss: 0.310057 | Validation loss: 0.464070
Epoch: 29 | Training loss: 0.305203 | Validation loss: 0.459905
Epoch: 30 | Training loss: 0.300554 | Validation loss: 0.455920
Epoch: 31 | Training loss: 0.296096 | Validation loss: 0.452127
Epoch: 32 | Training loss: 0.291808 | Validation loss: 0.448500
Epoch: 33 | Training loss: 0.287703 | Validation loss: 0.445029
Epoch: 34 | Training loss: 0.283741 | Validation loss: 0.441705
Epoch: 35 | Training loss: 0.279931 | Validation loss: 0.438493
Epoch: 36 | Training loss: 0.276256 | Validation loss: 0.435415
Epoch: 37 | Training loss: 0.272708 | Validation loss: 0.432458
Epoch: 38 | Training loss: 0.269280 | Validation loss: 0.429607
Epoch: 39 | Training loss: 0.265966 | Validation loss: 0.426877
Epoch: 40 | Training loss: 0.262757 | Validation loss: 0.424220
Epoch: 41 | Training loss: 0.259653 | Validation loss: 0.421674
Epoch: 42 | Training loss: 0.256642 | Validation loss: 0.419198
Epoch: 43 | Training loss: 0.253725 | Validation loss: 0.416828
Epoch: 44 | Training loss: 0.250892 | Validation loss: 0.414537
Epoch: 45 | Training loss: 0.248141 | Validation loss: 0.412309
Epoch: 46 | Training loss: 0.245468 | Validation loss: 0.410146
Epoch: 47 | Training loss: 0.242869 | Validation loss: 0.408079
Epoch: 48 | Training loss: 0.240341 | Validation loss: 0.406046
Epoch: 49 | Training loss: 0.237881 | Validation loss: 0.404072
Epoch: 50 | Training loss: 0.235484 | Validation loss: 0.402146
```

```
Epoch: 51 | Training loss: 0.233150 | Validation loss: 0.400296
Epoch: 52 | Training loss: 0.230874 | Validation loss: 0.398483
Epoch: 53 | Training loss: 0.228655 | Validation loss: 0.396735
Epoch: 54 | Training loss: 0.226488 | Validation loss: 0.395040
Epoch: 55 | Training loss: 0.224372 | Validation loss: 0.393373
Epoch: 56 | Training loss: 0.222308 | Validation loss: 0.391747
Epoch: 57 | Training loss: 0.220290 | Validation loss: 0.390183
Epoch: 58 | Training loss: 0.218317 | Validation loss: 0.388678
Epoch: 59 | Training loss: 0.216391 | Validation loss: 0.387191
Epoch: 60 | Training loss: 0.214506 | Validation loss: 0.385733
Epoch: 61 | Training loss: 0.212661 | Validation loss: 0.384317
Epoch: 62 | Training loss: 0.210855 | Validation loss: 0.382937
Epoch: 63 | Training loss: 0.209088 | Validation loss: 0.381568
Epoch: 64 | Training loss: 0.207354 | Validation loss: 0.380251
Epoch: 65 | Training loss: 0.205660 | Validation loss: 0.378945
Epoch: 66 | Training loss: 0.203994 | Validation loss: 0.377652
Epoch: 67 | Training loss: 0.202371 | Validation loss: 0.376424
Epoch: 68 | Training loss: 0.200771 | Validation loss: 0.375198
Epoch: 69 | Training loss: 0.199209 | Validation loss: 0.374035
Epoch: 70 | Training loss: 0.197672 | Validation loss: 0.372909
Epoch: 71 | Training loss: 0.196163 | Validation loss: 0.371799
Epoch: 72 | Training loss: 0.194681 | Validation loss: 0.370720
Epoch: 73 | Training loss: 0.193228 | Validation loss: 0.369628
Epoch: 74 | Training loss: 0.191803 | Validation loss: 0.368586
Epoch: 75 | Training loss: 0.190398 | Validation loss: 0.367553
Epoch: 76 | Training loss: 0.189024 | Validation loss: 0.366531
Epoch: 77 | Training loss: 0.187672 | Validation loss: 0.365546
lr = 0.001000, alpha= 0.001000; Predict_Class:0.827243
Epoch: 0 | Training loss: 0.990602 | Validation loss: 0.977745
Epoch: 1 | Training loss: 0.840381 | Validation loss: 0.895537
Epoch: 2 | Training loss: 0.747175 | Validation loss: 0.833714
Epoch: 3 | Training loss: 0.681331 | Validation loss: 0.785051
Epoch: 4 | Training loss: 0.631415 | Validation loss: 0.745609
Epoch: 5 | Training loss: 0.591891 | Validation loss: 0.713017
Epoch: 6 | Training loss: 0.559500 | Validation loss: 0.685475
Epoch: 7 | Training loss: 0.532220 | Validation loss: 0.661820
Epoch: 8 | Training loss: 0.508801 | Validation loss: 0.641125
```

```
Epoch: 9 | Training loss: 0.488391 | Validation loss: 0.622933
Epoch: 10 | Training loss: 0.470366 | Validation loss: 0.606820
Epoch: 11 | Training loss: 0.454259 | Validation loss: 0.592334
Epoch: 12 | Training loss: 0.439742 | Validation loss: 0.579290
Epoch: 13 | Training loss: 0.426555 | Validation loss: 0.567436
Epoch: 14 | Training loss: 0.414507 | Validation loss: 0.556571
Epoch: 15 | Training loss: 0.403429 | Validation loss: 0.546640
Epoch: 16 | Training loss: 0.393187 | Validation loss: 0.537505
Epoch: 17 | Training loss: 0.383688 | Validation loss: 0.528986
Epoch: 18 | Training loss: 0.374835 | Validation loss: 0.521064
Epoch: 19 | Training loss: 0.366554 | Validation loss: 0.513686
Epoch: 20 | Training loss: 0.358786 | Validation loss: 0.506807
Epoch: 21 | Training loss: 0.351480 | Validation loss: 0.500334
Epoch: 22 | Training loss: 0.344582 | Validation loss: 0.494234
Epoch: 23 | Training loss: 0.338066 | Validation loss: 0.488493
Epoch: 24 | Training loss: 0.331892 | Validation loss: 0.483100
Epoch: 25 | Training loss: 0.326026 | Validation loss: 0.477988
Epoch: 26 | Training loss: 0.320446 | Validation loss: 0.473141
Epoch: 27 | Training loss: 0.315127 | Validation loss: 0.468554
Epoch: 28 | Training loss: 0.310047 | Validation loss: 0.464165
Epoch: 29 | Training loss: 0.305193 | Validation loss: 0.459977
Epoch: 30 | Training loss: 0.300541 | Validation loss: 0.455997
Epoch: 31 | Training loss: 0.296083 | Validation loss: 0.452214
Epoch: 32 | Training loss: 0.291801 | Validation loss: 0.448618
Epoch: 33 | Training loss: 0.287690 | Validation loss: 0.445115
Epoch: 34 | Training loss: 0.283731 | Validation loss: 0.441778
Epoch: 35 | Training loss: 0.279917 | Validation loss: 0.438607
Epoch: 36 | Training loss: 0.276242 | Validation loss: 0.435522
Epoch: 37 | Training loss: 0.272686 | Validation loss: 0.432503
Epoch: 38 | Training loss: 0.269267 | Validation loss: 0.429671
Epoch: 39 | Training loss: 0.265954 | Validation loss: 0.426931
Epoch: 40 | Training loss: 0.262749 | Validation loss: 0.424274
Epoch: 41 | Training loss: 0.259644 | Validation loss: 0.421728
Epoch: 42 | Training loss: 0.256632 | Validation loss: 0.419247
Epoch: 43 | Training loss: 0.253716 | Validation loss: 0.416879
Epoch: 44 | Training loss: 0.250875 | Validation loss: 0.414529
Epoch: 45 | Training loss: 0.248134 | Validation loss: 0.412316
```

```
Epoch: 46 | Training loss: 0.245460 | Validation loss: 0.410173
Epoch: 47 | Training loss: 0.242860 | Validation loss: 0.408112
Epoch: 48 | Training loss: 0.240329 | Validation loss: 0.406081
Epoch: 49 | Training loss: 0.237871 | Validation loss: 0.404099
Epoch: 50 | Training loss: 0.235477 | Validation loss: 0.402190
Epoch: 51 | Training loss: 0.233141 | Validation loss: 0.400315
Epoch: 52 | Training loss: 0.230861 | Validation loss: 0.398472
Epoch: 53 | Training loss: 0.228646 | Validation loss: 0.396730
Epoch: 54 | Training loss: 0.226478 | Validation loss: 0.395014
Epoch: 55 | Training loss: 0.224364 | Validation loss: 0.393375
Epoch: 56 | Training loss: 0.222299 | Validation loss: 0.391754
Epoch: 57 | Training loss: 0.220282 | Validation loss: 0.390183
Epoch: 58 | Training loss: 0.218312 | Validation loss: 0.388654
Epoch: 59 | Training loss: 0.216383 | Validation loss: 0.387165
Epoch: 60 | Training loss: 0.214498 | Validation loss: 0.385720
Epoch: 61 | Training loss: 0.212653 | Validation loss: 0.384299
Epoch: 62 | Training loss: 0.210850 | Validation loss: 0.382920
Epoch: 63 | Training loss: 0.209081 | Validation loss: 0.381562
Epoch: 64 | Training loss: 0.207351 | Validation loss: 0.380262
Epoch: 65 | Training loss: 0.205654 | Validation loss: 0.378977
Epoch: 66 | Training loss: 0.203992 | Validation loss: 0.377725
Epoch: 67 | Training loss: 0.202363 | Validation loss: 0.376486
Epoch: 68 | Training loss: 0.200763 | Validation loss: 0.375290
Epoch: 69 | Training loss: 0.199198 | Validation loss: 0.374098
Epoch: 70 | Training loss: 0.197663 | Validation loss: 0.372941
Epoch: 71 | Training loss: 0.196156 | Validation loss: 0.371809
Epoch: 72 | Training loss: 0.194675 | Validation loss: 0.370703
Epoch: 73 | Training loss: 0.193223 | Validation loss: 0.369627
Epoch: 74 | Training loss: 0.191797 | Validation loss: 0.368581
Epoch: 75 | Training loss: 0.190396 | Validation loss: 0.367554
Epoch: 76 | Training loss: 0.189018 | Validation loss: 0.366540
Epoch: 77 | Training loss: 0.187664 | Validation loss: 0.365540
Epoch: 78 | Training loss: 0.186336 | Validation loss: 0.364574
lr = 0.001000, alpha= 0.000500; Predict_Class:0.825871
Epoch: 0 | Training loss: 0.990624 | Validation loss: 0.978114
Epoch: 1 | Training loss: 0.840351 | Validation loss: 0.895639
Epoch: 2 | Training loss: 0.747046 | Validation loss: 0.833517
```

```
Epoch: 3 | Training loss: 0.681165 | Validation loss: 0.784850
Epoch: 4 | Training loss: 0.631286 | Validation loss: 0.745488
Epoch: 5 | Training loss: 0.591803 | Validation loss: 0.712826
Epoch: 6 | Training loss: 0.559423 | Validation loss: 0.685282
Epoch: 7 | Training loss: 0.532150 | Validation loss: 0.661541
Epoch: 8 | Training loss: 0.508776 | Validation loss: 0.640966
Epoch: 9 | Training loss: 0.488375 | Validation loss: 0.622821
Epoch: 10 | Training loss: 0.470354 | Validation loss: 0.606717
Epoch: 11 | Training loss: 0.454250 | Validation loss: 0.592227
Epoch: 12 | Training loss: 0.439731 | Validation loss: 0.579159
Epoch: 13 | Training loss: 0.426543 | Validation loss: 0.567244
Epoch: 14 | Training loss: 0.414509 | Validation loss: 0.556433
Epoch: 15 | Training loss: 0.403433 | Validation loss: 0.546507
Epoch: 16 | Training loss: 0.393199 | Validation loss: 0.537325
Epoch: 17 | Training loss: 0.383697 | Validation loss: 0.528842
Epoch: 18 | Training loss: 0.374851 | Validation loss: 0.520943
Epoch: 19 | Training loss: 0.366564 | Validation loss: 0.513530
Epoch: 20 | Training loss: 0.358802 | Validation loss: 0.506639
Epoch: 21 | Training loss: 0.351496 | Validation loss: 0.500190
Epoch: 22 | Training loss: 0.344606 | Validation loss: 0.494132
Epoch: 23 | Training loss: 0.338074 | Validation loss: 0.488374
Epoch: 24 | Training loss: 0.331905 | Validation loss: 0.482999
Epoch: 25 | Training loss: 0.326041 | Validation loss: 0.477916
Epoch: 26 | Training loss: 0.320458 | Validation loss: 0.473087
Epoch: 27 | Training loss: 0.315139 | Validation loss: 0.468505
Epoch: 28 | Training loss: 0.310059 | Validation loss: 0.464118
Epoch: 29 | Training loss: 0.305206 | Validation loss: 0.459956
Epoch: 30 | Training loss: 0.300554 | Validation loss: 0.455970
Epoch: 31 | Training loss: 0.296095 | Validation loss: 0.452166
Epoch: 32 | Training loss: 0.291814 | Validation loss: 0.448550
Epoch: 33 | Training loss: 0.287701 | Validation loss: 0.445067
Epoch: 34 | Training loss: 0.283743 | Validation loss: 0.441729
Epoch: 35 | Training loss: 0.279931 | Validation loss: 0.438541
Epoch: 36 | Training loss: 0.276255 | Validation loss: 0.435483
Epoch: 37 | Training loss: 0.272703 | Validation loss: 0.432507
Epoch: 38 | Training loss: 0.269279 | Validation loss: 0.429660
Epoch: 39 | Training loss: 0.265962 | Validation loss: 0.426897
```

```
Epoch: 40 | Training loss: 0.262756 | Validation loss: 0.424255
Epoch: 41 | Training loss: 0.259651 | Validation loss: 0.421696
Epoch: 42 | Training loss: 0.256642 | Validation loss: 0.419235
Epoch: 43 | Training loss: 0.253724 | Validation loss: 0.416840
Epoch: 44 | Training loss: 0.250891 | Validation loss: 0.414531
Epoch: 45 | Training loss: 0.248137 | Validation loss: 0.412267
Epoch: 46 | Training loss: 0.245468 | Validation loss: 0.410108
Epoch: 47 | Training loss: 0.242871 | Validation loss: 0.408030
Epoch: 48 | Training loss: 0.240343 | Validation loss: 0.406017
Epoch: 49 | Training loss: 0.237879 | Validation loss: 0.404054
Epoch: 50 | Training loss: 0.235485 | Validation loss: 0.402156
Epoch: 51 | Training loss: 0.233149 | Validation loss: 0.400306
Epoch: 52 | Training loss: 0.230872 | Validation loss: 0.398512
Epoch: 53 | Training loss: 0.228652 | Validation loss: 0.396762
Epoch: 54 | Training loss: 0.226486 | Validation loss: 0.395064
Epoch: 55 | Training loss: 0.224372 | Validation loss: 0.393414
Epoch: 56 | Training loss: 0.222303 | Validation loss: 0.391805
Epoch: 57 | Training loss: 0.220290 | Validation loss: 0.390232
Epoch: 58 | Training loss: 0.218316 | Validation loss: 0.388716
Epoch: 59 | Training loss: 0.216390 | Validation loss: 0.387208
Epoch: 60 | Training loss: 0.214501 | Validation loss: 0.385718
Epoch: 61 | Training loss: 0.212660 | Validation loss: 0.384310
Epoch: 62 | Training loss: 0.210848 | Validation loss: 0.382895
Epoch: 63 | Training loss: 0.209088 | Validation loss: 0.381551
Epoch: 64 | Training loss: 0.207357 | Validation loss: 0.380252
Epoch: 65 | Training loss: 0.205657 | Validation loss: 0.378976
Epoch: 66 | Training loss: 0.203998 | Validation loss: 0.377726
Epoch: 67 | Training loss: 0.202368 | Validation loss: 0.376496
Epoch: 68 | Training loss: 0.200772 | Validation loss: 0.375307
Epoch: 69 | Training loss: 0.199205 | Validation loss: 0.374157
Epoch: 70 | Training loss: 0.197669 | Validation loss: 0.373007
Epoch: 71 | Training loss: 0.196161 | Validation loss: 0.371868
Epoch: 72 | Training loss: 0.194679 | Validation loss: 0.370757
Epoch: 73 | Training loss: 0.193228 | Validation loss: 0.369690
Epoch: 74 | Training loss: 0.191801 | Validation loss: 0.368638
Epoch: 75 | Training loss: 0.190399 | Validation loss: 0.367612
Epoch: 76 | Training loss: 0.189023 | Validation loss: 0.366593
```

```
Epoch: 77 | Training loss: 0.187670 | Validation loss: 0.365601
lr = 0.001000, alpha= 0.000100; Predict_Class:0.827243
Epoch: 0 | Training loss: 1.036428 | Validation loss: 1.030935
Epoch: 1 | Training loss: 0.937166 | Validation loss: 0.978131
Epoch: 2 | Training loss: 0.865730 | Validation loss: 0.933858
Epoch: 3 | Training loss: 0.809748 | Validation loss: 0.895964
Epoch: 4 | Training loss: 0.764142 | Validation loss: 0.862914
Epoch: 5 | Training loss: 0.726043 | Validation loss: 0.833930
Epoch: 6 | Training loss: 0.693571 | Validation loss: 0.808239
Epoch: 7 | Training loss: 0.665479 | Validation loss: 0.785257
Epoch: 8 | Training loss: 0.640838 | Validation loss: 0.764574
Epoch: 9 | Training loss: 0.618968 | Validation loss: 0.745874
Epoch: 10 | Training loss: 0.599390 | Validation loss: 0.728837
Epoch: 11 | Training loss: 0.581705 | Validation loss: 0.713242
Epoch: 12 | Training loss: 0.565625 | Validation loss: 0.698921
Epoch: 13 | Training loss: 0.550912 | Validation loss: 0.685682
Epoch: 14 | Training loss: 0.537370 | Validation loss: 0.673415
Epoch: 15 | Training loss: 0.524845 | Validation loss: 0.661996
Epoch: 16 | Training loss: 0.513212 | Validation loss: 0.651357
Epoch: 17 | Training loss: 0.502361 | Validation loss: 0.641390
Epoch: 18 | Training loss: 0.492206 | Validation loss: 0.632025
Epoch: 19 | Training loss: 0.482674 | Validation loss: 0.623212
Epoch: 20 | Training loss: 0.473697 | Validation loss: 0.614907
Epoch: 21 | Training loss: 0.465223 | Validation loss: 0.607072
Epoch: 22 | Training loss: 0.457203 | Validation loss: 0.599632
Epoch: 23 | Training loss: 0.449596 | Validation loss: 0.592581
Epoch: 24 | Training loss: 0.442369 | Validation loss: 0.585893
Epoch: 25 | Training loss: 0.435484 | Validation loss: 0.579515
Epoch: 26 | Training loss: 0.428920 | Validation loss: 0.573444
Epoch: 27 | Training loss: 0.422646 | Validation loss: 0.567651
Epoch: 28 | Training loss: 0.416641 | Validation loss: 0.562120
Epoch: 29 | Training loss: 0.410890 | Validation loss: 0.556818
Epoch: 30 | Training loss: 0.405368 | Validation loss: 0.551737
Epoch: 31 | Training loss: 0.400064 | Validation loss: 0.546861
Epoch: 32 | Training loss: 0.394961 | Validation loss: 0.542181
Epoch: 33 | Training loss: 0.390050 | Validation loss: 0.537682
Epoch: 34 | Training loss: 0.385314 | Validation loss: 0.533345
```

```
Epoch: 35 | Training loss: 0.380743 | Validation loss: 0.529179
Epoch: 36 | Training loss: 0.376331 | Validation loss: 0.525155
Epoch: 37 | Training loss: 0.372064 | Validation loss: 0.521264
Epoch: 38 | Training loss: 0.367936 | Validation loss: 0.517512
Epoch: 39 | Training loss: 0.363938 | Validation loss: 0.513890
Epoch: 40 | Training loss: 0.360065 | Validation loss: 0.510386
Epoch: 41 | Training loss: 0.356312 | Validation loss: 0.506993
Epoch: 42 | Training loss: 0.352668 | Validation loss: 0.503701
Epoch: 43 | Training loss: 0.349129 | Validation loss: 0.500518
Epoch: 44 | Training loss: 0.345694 | Validation loss: 0.497432
Epoch: 45 | Training loss: 0.342351 | Validation loss: 0.494437
Epoch: 46 | Training loss: 0.339101 | Validation loss: 0.491528
Epoch: 47 | Training loss: 0.335938 | Validation loss: 0.488708
Epoch: 48 | Training loss: 0.332858 | Validation loss: 0.485962
Epoch: 49 | Training loss: 0.329856 | Validation loss: 0.483300
Epoch: 50 | Training loss: 0.326931 | Validation loss: 0.480705
Epoch: 51 | Training loss: 0.324077 | Validation loss: 0.478176
Epoch: 52 | Training loss: 0.321293 | Validation loss: 0.475712
Epoch: 53 | Training loss: 0.318577 | Validation loss: 0.473318
Epoch: 54 | Training loss: 0.315924 | Validation loss: 0.470982
Epoch: 55 | Training loss: 0.313333 | Validation loss: 0.468712
Epoch: 56 | Training loss: 0.310800 | Validation loss: 0.466500
Epoch: 57 | Training loss: 0.308323 | Validation loss: 0.464339
Epoch: 58 | Training loss: 0.305900 | Validation loss: 0.462221
Epoch: 59 | Training loss: 0.303532 | Validation loss: 0.460162
Epoch: 60 | Training loss: 0.301212 | Validation loss: 0.458153
Epoch: 61 | Training loss: 0.298942 | Validation loss: 0.456184
Epoch: 62 | Training loss: 0.296718 | Validation loss: 0.454262
Epoch: 63 | Training loss: 0.294541 | Validation loss: 0.452380
Epoch: 64 | Training loss: 0.292406 | Validation loss: 0.450539
Epoch: 65 | Training loss: 0.290312 | Validation loss: 0.448733
Epoch: 66 | Training loss: 0.288261 | Validation loss: 0.446969
Epoch: 67 | Training loss: 0.286249 | Validation loss: 0.445241
Epoch: 68 | Training loss: 0.284275 | Validation loss: 0.443552
Epoch: 69 | Training loss: 0.282338 | Validation loss: 0.441900
Epoch: 70 | Training loss: 0.280436 | Validation loss: 0.440286
Epoch: 71 | Training loss: 0.278568 | Validation loss: 0.438695
```

```
Epoch: 72 | Training loss: 0.276735 | Validation loss: 0.437139
Epoch: 73 | Training loss: 0.274934 | Validation loss: 0.435620
Epoch: 74 | Training loss: 0.273164 | Validation loss: 0.434120
Epoch: 75 | Training loss: 0.271424 | Validation loss: 0.432649
Epoch: 76 | Training loss: 0.269715 | Validation loss: 0.431214
Epoch: 77 | Training loss: 0.268032 | Validation loss: 0.429789
Epoch: 78 | Training loss: 0.266381 | Validation loss: 0.428409
Epoch: 79 | Training loss: 0.264754 | Validation loss: 0.427049
Epoch: 80 | Training loss: 0.263155 | Validation loss: 0.425723
Epoch: 81 | Training loss: 0.261580 | Validation loss: 0.424418
Epoch: 82 | Training loss: 0.260031 | Validation loss: 0.423127
Epoch: 83 | Training loss: 0.258506 | Validation loss: 0.421854
Epoch: 84 | Training loss: 0.257004 | Validation loss: 0.420597
Epoch: 85 | Training loss: 0.255527 | Validation loss: 0.419371
Epoch: 86 | Training loss: 0.254071 | Validation loss: 0.418169
Epoch: 87 | Training loss: 0.252637 | Validation loss: 0.416989
Epoch: 88 | Training loss: 0.251224 | Validation loss: 0.415821
Epoch: 89 | Training loss: 0.249831 | Validation loss: 0.414678
Epoch: 90 | Training loss: 0.248459 | Validation loss: 0.413542
Epoch: 91 | Training loss: 0.247107 | Validation loss: 0.412434
Epoch: 92 | Training loss: 0.245774 | Validation loss: 0.411348
Epoch: 93 | Training loss: 0.244460 | Validation loss: 0.410278
Epoch: 94 | Training loss: 0.243163 | Validation loss: 0.409224
Epoch: 95 | Training loss: 0.241884 | Validation loss: 0.408180
Epoch: 96 | Training loss: 0.240623 | Validation loss: 0.407164
Epoch: 97 | Training loss: 0.239379 | Validation loss: 0.406154
Epoch: 98 | Training loss: 0.238150 | Validation loss: 0.405153
Epoch: 99 | Training loss: 0.236940 | Validation loss: 0.404181
lr = 0.000500, alpha= 0.001000; Predict_Class:0.831947
Epoch: 0 | Training loss: 1.036607 | Validation loss: 1.031025
Epoch: 1 | Training loss: 0.937346 | Validation loss: 0.978162
Epoch: 2 | Training loss: 0.865740 | Validation loss: 0.933890
Epoch: 3 | Training loss: 0.809700 | Validation loss: 0.895950
Epoch: 4 | Training loss: 0.764098 | Validation loss: 0.862943
Epoch: 5 | Training loss: 0.726014 | Validation loss: 0.833947
Epoch: 6 | Training loss: 0.693564 | Validation loss: 0.808250
Epoch: 7 | Training loss: 0.665469 | Validation loss: 0.785245
```

```
Epoch: 8 | Training loss: 0.640832 | Validation loss: 0.764589
Epoch: 9 | Training loss: 0.618970 | Validation loss: 0.745884
Epoch: 10 | Training loss: 0.599379 | Validation loss: 0.728839
Epoch: 11 | Training loss: 0.581700 | Validation loss: 0.713229
Epoch: 12 | Training loss: 0.565614 | Validation loss: 0.698886
Epoch: 13 | Training loss: 0.550894 | Validation loss: 0.685649
Epoch: 14 | Training loss: 0.537349 | Validation loss: 0.673373
Epoch: 15 | Training loss: 0.524820 | Validation loss: 0.661959
Epoch: 16 | Training loss: 0.513190 | Validation loss: 0.651314
Epoch: 17 | Training loss: 0.502345 | Validation loss: 0.641350
Epoch: 18 | Training loss: 0.492196 | Validation loss: 0.631998
Epoch: 19 | Training loss: 0.482665 | Validation loss: 0.623191
Epoch: 20 | Training loss: 0.473688 | Validation loss: 0.614889
Epoch: 21 | Training loss: 0.465215 | Validation loss: 0.607053
Epoch: 22 | Training loss: 0.457197 | Validation loss: 0.599628
Epoch: 23 | Training loss: 0.449594 | Validation loss: 0.592582
Epoch: 24 | Training loss: 0.442365 | Validation loss: 0.585885
Epoch: 25 | Training loss: 0.435479 | Validation loss: 0.579505
Epoch: 26 | Training loss: 0.428915 | Validation loss: 0.573437
Epoch: 27 | Training loss: 0.422642 | Validation loss: 0.567640
Epoch: 28 | Training loss: 0.416638 | Validation loss: 0.562104
Epoch: 29 | Training loss: 0.410887 | Validation loss: 0.556794
Epoch: 30 | Training loss: 0.405368 | Validation loss: 0.551712
Epoch: 31 | Training loss: 0.400063 | Validation loss: 0.546829
Epoch: 32 | Training loss: 0.394959 | Validation loss: 0.542142
Epoch: 33 | Training loss: 0.390046 | Validation loss: 0.537637
Epoch: 34 | Training loss: 0.385312 | Validation loss: 0.533313
Epoch: 35 | Training loss: 0.380741 | Validation loss: 0.529135
Epoch: 36 | Training loss: 0.376329 | Validation loss: 0.525112
Epoch: 37 | Training loss: 0.372062 | Validation loss: 0.521240
Epoch: 38 | Training loss: 0.367934 | Validation loss: 0.517491
Epoch: 39 | Training loss: 0.363937 | Validation loss: 0.513870
Epoch: 40 | Training loss: 0.360065 | Validation loss: 0.510361
Epoch: 41 | Training loss: 0.356310 | Validation loss: 0.506973
Epoch: 42 | Training loss: 0.352667 | Validation loss: 0.503691
Epoch: 43 | Training loss: 0.349130 | Validation loss: 0.500506
Epoch: 44 | Training loss: 0.345693 | Validation loss: 0.497424
```

```
Epoch: 45 | Training loss: 0.342352 | Validation loss: 0.494433
Epoch: 46 | Training loss: 0.339102 | Validation loss: 0.491528
Epoch: 47 | Training loss: 0.335938 | Validation loss: 0.488702
Epoch: 48 | Training loss: 0.332858 | Validation loss: 0.485956
Epoch: 49 | Training loss: 0.329858 | Validation loss: 0.483297
Epoch: 50 | Training loss: 0.326931 | Validation loss: 0.480699
Epoch: 51 | Training loss: 0.324078 | Validation loss: 0.478177
Epoch: 52 | Training loss: 0.321292 | Validation loss: 0.475705
Epoch: 53 | Training loss: 0.318579 | Validation loss: 0.473316
Epoch: 54 | Training loss: 0.315924 | Validation loss: 0.470977
Epoch: 55 | Training loss: 0.313334 | Validation loss: 0.468708
Epoch: 56 | Training loss: 0.310800 | Validation loss: 0.466493
Epoch: 57 | Training loss: 0.308325 | Validation loss: 0.464332
Epoch: 58 | Training loss: 0.305903 | Validation loss: 0.462223
Epoch: 59 | Training loss: 0.303533 | Validation loss: 0.460165
Epoch: 60 | Training loss: 0.301214 | Validation loss: 0.458150
Epoch: 61 | Training loss: 0.298943 | Validation loss: 0.456184
Epoch: 62 | Training loss: 0.296720 | Validation loss: 0.454257
Epoch: 63 | Training loss: 0.294540 | Validation loss: 0.452369
Epoch: 64 | Training loss: 0.292407 | Validation loss: 0.450526
Epoch: 65 | Training loss: 0.290314 | Validation loss: 0.448725
Epoch: 66 | Training loss: 0.288262 | Validation loss: 0.446961
Epoch: 67 | Training loss: 0.286251 | Validation loss: 0.445241
Epoch: 68 | Training loss: 0.284276 | Validation loss: 0.443550
Epoch: 69 | Training loss: 0.282338 | Validation loss: 0.441905
Epoch: 70 | Training loss: 0.280437 | Validation loss: 0.440292
Epoch: 71 | Training loss: 0.278569 | Validation loss: 0.438712
Epoch: 72 | Training loss: 0.276736 | Validation loss: 0.437157
Epoch: 73 | Training loss: 0.274935 | Validation loss: 0.435635
Epoch: 74 | Training loss: 0.273165 | Validation loss: 0.434137
Epoch: 75 | Training loss: 0.271425 | Validation loss: 0.432678
Epoch: 76 | Training loss: 0.269716 | Validation loss: 0.431237
Epoch: 77 | Training loss: 0.268034 | Validation loss: 0.429823
Epoch: 78 | Training loss: 0.266380 | Validation loss: 0.428440
Epoch: 79 | Training loss: 0.264755 | Validation loss: 0.427073
Epoch: 80 | Training loss: 0.263156 | Validation loss: 0.425735
Epoch: 81 | Training loss: 0.261582 | Validation loss: 0.424422
```

```
Epoch: 82 | Training loss: 0.260033 | Validation loss: 0.423132
Epoch: 83 | Training loss: 0.258507 | Validation loss: 0.421858
Epoch: 84 | Training loss: 0.257007 | Validation loss: 0.420613
Epoch: 85 | Training loss: 0.255528 | Validation loss: 0.419391
Epoch: 86 | Training loss: 0.254072 | Validation loss: 0.418183
Epoch: 87 | Training loss: 0.252638 | Validation loss: 0.417001
Epoch: 88 | Training loss: 0.251224 | Validation loss: 0.415832
Epoch: 89 | Training loss: 0.249832 | Validation loss: 0.414685
Epoch: 90 | Training loss: 0.248461 | Validation loss: 0.413561
Epoch: 91 | Training loss: 0.247108 | Validation loss: 0.412458
Epoch: 92 | Training loss: 0.245775 | Validation loss: 0.411364
Epoch: 93 | Training loss: 0.244460 | Validation loss: 0.410293
Epoch: 94 | Training loss: 0.243164 | Validation loss: 0.409241
Epoch: 95 | Training loss: 0.241885 | Validation loss: 0.408204
Epoch: 96 | Training loss: 0.240624 | Validation loss: 0.407183
Epoch: 97 | Training loss: 0.239380 | Validation loss: 0.406177
Epoch: 98 | Training loss: 0.238153 | Validation loss: 0.405182
lr = 0.000500, alpha= 0.000500; Predict_Class:0.831947
Epoch: 0 | Training loss: 1.036498 | Validation loss: 1.031044
Epoch: 1 | Training loss: 0.937248 | Validation loss: 0.978159
Epoch: 2 | Training loss: 0.865804 | Validation loss: 0.933931
Epoch: 3 | Training loss: 0.809795 | Validation loss: 0.895972
Epoch: 4 | Training loss: 0.764150 | Validation loss: 0.862927
Epoch: 5 | Training loss: 0.726047 | Validation loss: 0.833944
Epoch: 6 | Training loss: 0.693583 | Validation loss: 0.808211
Epoch: 7 | Training loss: 0.665485 | Validation loss: 0.785238
Epoch: 8 | Training loss: 0.640833 | Validation loss: 0.764583
Epoch: 9 | Training loss: 0.618958 | Validation loss: 0.745870
Epoch: 10 | Training loss: 0.599385 | Validation loss: 0.728836
Epoch: 11 | Training loss: 0.581704 | Validation loss: 0.713238
Epoch: 12 | Training loss: 0.565631 | Validation loss: 0.698915
Epoch: 13 | Training loss: 0.550911 | Validation loss: 0.685677
Epoch: 14 | Training loss: 0.537360 | Validation loss: 0.673395
Epoch: 15 | Training loss: 0.524832 | Validation loss: 0.661974
Epoch: 16 | Training loss: 0.513192 | Validation loss: 0.651311
Epoch: 17 | Training loss: 0.502344 | Validation loss: 0.641344
Epoch: 18 | Training loss: 0.492190 | Validation loss: 0.632003
```

```
Epoch: 19 | Training loss: 0.482661 | Validation loss: 0.623207
Epoch: 20 | Training loss: 0.473686 | Validation loss: 0.614913
Epoch: 21 | Training loss: 0.465215 | Validation loss: 0.607076
Epoch: 22 | Training loss: 0.457196 | Validation loss: 0.599640
Epoch: 23 | Training loss: 0.449588 | Validation loss: 0.592583
Epoch: 24 | Training loss: 0.442361 | Validation loss: 0.585888
Epoch: 25 | Training loss: 0.435476 | Validation loss: 0.579511
Epoch: 26 | Training loss: 0.428911 | Validation loss: 0.573434
Epoch: 27 | Training loss: 0.422638 | Validation loss: 0.567633
Epoch: 28 | Training loss: 0.416632 | Validation loss: 0.562091
Epoch: 29 | Training loss: 0.410880 | Validation loss: 0.556786
Epoch: 30 | Training loss: 0.405360 | Validation loss: 0.551696
Epoch: 31 | Training loss: 0.400057 | Validation loss: 0.546828
Epoch: 32 | Training loss: 0.394954 | Validation loss: 0.542136
Epoch: 33 | Training loss: 0.390043 | Validation loss: 0.537635
Epoch: 34 | Training loss: 0.385308 | Validation loss: 0.533303
Epoch: 35 | Training loss: 0.380738 | Validation loss: 0.529142
Epoch: 36 | Training loss: 0.376326 | Validation loss: 0.525120
Epoch: 37 | Training loss: 0.372061 | Validation loss: 0.521237
Epoch: 38 | Training loss: 0.367932 | Validation loss: 0.517484
Epoch: 39 | Training loss: 0.363936 | Validation loss: 0.513856
Epoch: 40 | Training loss: 0.360062 | Validation loss: 0.510358
Epoch: 41 | Training loss: 0.356309 | Validation loss: 0.506968
Epoch: 42 | Training loss: 0.352665 | Validation loss: 0.503681
Epoch: 43 | Training loss: 0.349128 | Validation loss: 0.500499
Epoch: 44 | Training loss: 0.345690 | Validation loss: 0.497410
Epoch: 45 | Training loss: 0.342349 | Validation loss: 0.494420
Epoch: 46 | Training loss: 0.339098 | Validation loss: 0.491517
Epoch: 47 | Training loss: 0.335934 | Validation loss: 0.488701
Epoch: 48 | Training loss: 0.332854 | Validation loss: 0.485968
Epoch: 49 | Training loss: 0.329853 | Validation loss: 0.483293
Epoch: 50 | Training loss: 0.326928 | Validation loss: 0.480701
Epoch: 51 | Training loss: 0.324074 | Validation loss: 0.478169
Epoch: 52 | Training loss: 0.321293 | Validation loss: 0.475715
Epoch: 53 | Training loss: 0.318576 | Validation loss: 0.473318
Epoch: 54 | Training loss: 0.315923 | Validation loss: 0.470984
Epoch: 55 | Training loss: 0.313331 | Validation loss: 0.468714
```

```
Epoch: 56 | Training loss: 0.310798 | Validation loss: 0.466500
Epoch: 57 | Training loss: 0.308322 | Validation loss: 0.464336
Epoch: 58 | Training loss: 0.305897 | Validation loss: 0.462209
Epoch: 59 | Training loss: 0.303530 | Validation loss: 0.460148
Epoch: 60 | Training loss: 0.301211 | Validation loss: 0.458135
Epoch: 61 | Training loss: 0.298942 | Validation loss: 0.456168
Epoch: 62 | Training loss: 0.296718 | Validation loss: 0.454242
Epoch: 63 | Training loss: 0.294539 | Validation loss: 0.452358
Epoch: 64 | Training loss: 0.292405 | Validation loss: 0.450519
Epoch: 65 | Training loss: 0.290310 | Validation loss: 0.448707
Epoch: 66 | Training loss: 0.288261 | Validation loss: 0.446949
Epoch: 67 | Training loss: 0.286248 | Validation loss: 0.445221
Epoch: 68 | Training loss: 0.284274 | Validation loss: 0.443537
Epoch: 69 | Training loss: 0.282337 | Validation loss: 0.441883
Epoch: 70 | Training loss: 0.280433 | Validation loss: 0.440252
Epoch: 71 | Training loss: 0.278569 | Validation loss: 0.438670
Epoch: 72 | Training loss: 0.276734 | Validation loss: 0.437121
Epoch: 73 | Training loss: 0.274933 | Validation loss: 0.435594
Epoch: 74 | Training loss: 0.273163 | Validation loss: 0.434101
Epoch: 75 | Training loss: 0.271424 | Validation loss: 0.432635
Epoch: 76 | Training loss: 0.269714 | Validation loss: 0.431194
Epoch: 77 | Training loss: 0.268033 | Validation loss: 0.429788
Epoch: 78 | Training loss: 0.266380 | Validation loss: 0.428401
Epoch: 79 | Training loss: 0.264754 | Validation loss: 0.427046
Epoch: 80 | Training loss: 0.263154 | Validation loss: 0.425708
Epoch: 81 | Training loss: 0.261581 | Validation loss: 0.424396
Epoch: 82 | Training loss: 0.260031 | Validation loss: 0.423106
Epoch: 83 | Training loss: 0.258506 | Validation loss: 0.421835
Epoch: 84 | Training loss: 0.257005 | Validation loss: 0.420589
Epoch: 85 | Training loss: 0.255526 | Validation loss: 0.419363
Epoch: 86 | Training loss: 0.254071 | Validation loss: 0.418162
Epoch: 87 | Training loss: 0.252637 | Validation loss: 0.416987
Epoch: 88 | Training loss: 0.251224 | Validation loss: 0.415823
Epoch: 89 | Training loss: 0.249832 | Validation loss: 0.414680
Epoch: 90 | Training loss: 0.248459 | Validation loss: 0.413554
Epoch: 91 | Training loss: 0.247107 | Validation loss: 0.412450
Epoch: 92 | Training loss: 0.245774 | Validation loss: 0.411363
```

```
Epoch: 93 | Training loss: 0.244459 | Validation loss: 0.410290
Epoch: 94 | Training loss: 0.243163 | Validation loss: 0.409236
Epoch: 95 | Training loss: 0.241884 | Validation loss: 0.408195
Epoch: 96 | Training loss: 0.240623 | Validation loss: 0.407169
Epoch: 97 | Training loss: 0.239379 | Validation loss: 0.406163
Epoch: 98 | Training loss: 0.238151 | Validation loss: 0.405168
lr = 0.000500, alpha= 0.000100; Predict_Class:0.831947
Epoch: 0 | Training loss: 1.084223 | Validation loss: 1.083286
Epoch: 1 | Training loss: 1.056015 | Validation loss: 1.069044
Epoch: 2 | Training loss: 1.030895 | Validation loss: 1.055693
Epoch: 3 | Training loss: 1.008225 | Validation loss: 1.043099
Epoch: 4 | Training loss: 0.987513 | Validation loss: 1.031132
Epoch: 5 | Training loss: 0.968375 | Validation loss: 1.019709
Epoch: 6 | Training loss: 0.950555 | Validation loss: 1.008769
Epoch: 7 | Training loss: 0.933864 | Validation loss: 0.998255
Epoch: 8 | Training loss: 0.918150 | Validation loss: 0.988134
Epoch: 9 | Training loss: 0.903300 | Validation loss: 0.978371
Epoch: 10 | Training loss: 0.889233 | Validation loss: 0.968942
Epoch: 11 | Training loss: 0.875874 | Validation loss: 0.959829
Epoch: 12 | Training loss: 0.863159 | Validation loss: 0.951005
Epoch: 13 | Training loss: 0.851034 | Validation loss: 0.942460
Epoch: 14 | Training loss: 0.839452 | Validation loss: 0.934177
Epoch: 15 | Training loss: 0.828372 | Validation loss: 0.926142
Epoch: 16 | Training loss: 0.817758 | Validation loss: 0.918344
Epoch: 17 | Training loss: 0.807578 | Validation loss: 0.910768
Epoch: 18 | Training loss: 0.797804 | Validation loss: 0.903410
Epoch: 19 | Training loss: 0.788408 | Validation loss: 0.896256
Epoch: 20 | Training loss: 0.779368 | Validation loss: 0.889300
Epoch: 21 | Training loss: 0.770660 | Validation loss: 0.882531
Epoch: 22 | Training loss: 0.762265 | Validation loss: 0.875942
Epoch: 23 | Training loss: 0.754161 | Validation loss: 0.869525
Epoch: 24 | Training loss: 0.746337 | Validation loss: 0.863275
Epoch: 25 | Training loss: 0.738775 | Validation loss: 0.857184
Epoch: 26 | Training loss: 0.731461 | Validation loss: 0.851246
Epoch: 27 | Training loss: 0.724381 | Validation loss: 0.845454
Epoch: 28 | Training loss: 0.717522 | Validation loss: 0.839803
Epoch: 29 | Training loss: 0.710873 | Validation loss: 0.834289
```

```
Epoch: 30 | Training loss: 0.704425 | Validation loss: 0.828904
Epoch: 31 | Training loss: 0.698165 | Validation loss: 0.823647
Epoch: 32 | Training loss: 0.692086 | Validation loss: 0.818512
Epoch: 33 | Training loss: 0.686179 | Validation loss: 0.813492
Epoch: 34 | Training loss: 0.680435 | Validation loss: 0.808586
Epoch: 35 | Training loss: 0.674848 | Validation loss: 0.803788
Epoch: 36 | Training loss: 0.669410 | Validation loss: 0.799097
Epoch: 37 | Training loss: 0.664114 | Validation loss: 0.794507
Epoch: 38 | Training loss: 0.658955 | Validation loss: 0.790015
Epoch: 39 | Training loss: 0.653926 | Validation loss: 0.785618
Epoch: 40 | Training loss: 0.649023 | Validation loss: 0.781314
Epoch: 41 | Training loss: 0.644238 | Validation loss: 0.777097
Epoch: 42 | Training loss: 0.639568 | Validation loss: 0.772966
Epoch: 43 | Training loss: 0.635009 | Validation loss: 0.768919
Epoch: 44 | Training loss: 0.630555 | Validation loss: 0.764952
Epoch: 45 | Training loss: 0.626203 | Validation loss: 0.761062
Epoch: 46 | Training loss: 0.621947 | Validation loss: 0.757248
Epoch: 47 | Training loss: 0.617786 | Validation loss: 0.753507
Epoch: 48 | Training loss: 0.613715 | Validation loss: 0.749837
Epoch: 49 | Training loss: 0.609732 | Validation loss: 0.746235
Epoch: 50 | Training loss: 0.605832 | Validation loss: 0.742701
Epoch: 51 | Training loss: 0.602013 | Validation loss: 0.739232
Epoch: 52 | Training loss: 0.598272 | Validation loss: 0.735824
Epoch: 53 | Training loss: 0.594606 | Validation loss: 0.732478
Epoch: 54 | Training loss: 0.591014 | Validation loss: 0.729192
Epoch: 55 | Training loss: 0.587492 | Validation loss: 0.725963
Epoch: 56 | Training loss: 0.584037 | Validation loss: 0.722790
Epoch: 57 | Training loss: 0.580648 | Validation loss: 0.719671
Epoch: 58 | Training loss: 0.577322 | Validation loss: 0.716605
Epoch: 59 | Training loss: 0.574057 | Validation loss: 0.713590
Epoch: 60 | Training loss: 0.570853 | Validation loss: 0.710626
Epoch: 61 | Training loss: 0.567705 | Validation loss: 0.707710
Epoch: 62 | Training loss: 0.564615 | Validation loss: 0.704843
Epoch: 63 | Training loss: 0.561578 | Validation loss: 0.702021
Epoch: 64 | Training loss: 0.558593 | Validation loss: 0.699244
Epoch: 65 | Training loss: 0.555660 | Validation loss: 0.696511
Epoch: 66 | Training loss: 0.552776 | Validation loss: 0.693821
```

```
Epoch: 67 | Training loss: 0.549940 | Validation loss: 0.691173
Epoch: 68 | Training loss: 0.547151 | Validation loss: 0.688565
Epoch: 69 | Training loss: 0.544407 | Validation loss: 0.685996
Epoch: 70 | Training loss: 0.541707 | Validation loss: 0.683467
Epoch: 71 | Training loss: 0.539050 | Validation loss: 0.680975
Epoch: 72 | Training loss: 0.536434 | Validation loss: 0.678520
Epoch: 73 | Training loss: 0.533859 | Validation loss: 0.676100
Epoch: 74 | Training loss: 0.531323 | Validation loss: 0.673716
Epoch: 75 | Training loss: 0.528825 | Validation loss: 0.671366
Epoch: 76 | Training loss: 0.526365 | Validation loss: 0.669050
Epoch: 77 | Training loss: 0.523942 | Validation loss: 0.666766
Epoch: 78 | Training loss: 0.521553 | Validation loss: 0.664513
Epoch: 79 | Training loss: 0.519200 | Validation loss: 0.662292
Epoch: 80 | Training loss: 0.516880 | Validation loss: 0.660101
Epoch: 81 | Training loss: 0.514593 | Validation loss: 0.657941
Epoch: 82 | Training loss: 0.512338 | Validation loss: 0.655809
Epoch: 83 | Training loss: 0.510114 | Validation loss: 0.653707
Epoch: 84 | Training loss: 0.507921 | Validation loss: 0.651632
Epoch: 85 | Training loss: 0.505757 | Validation loss: 0.649584
Epoch: 86 | Training loss: 0.503623 | Validation loss: 0.647563
Epoch: 87 | Training loss: 0.501516 | Validation loss: 0.645568
Epoch: 88 | Training loss: 0.499438 | Validation loss: 0.643599
Epoch: 89 | Training loss: 0.497386 | Validation loss: 0.641654
Epoch: 90 | Training loss: 0.495361 | Validation loss: 0.639735
Epoch: 91 | Training loss: 0.493362 | Validation loss: 0.637839
Epoch: 92 | Training loss: 0.491389 | Validation loss: 0.635967
Epoch: 93 | Training loss: 0.489440 | Validation loss: 0.634117
Epoch: 94 | Training loss: 0.487515 | Validation loss: 0.632291
Epoch: 95 | Training loss: 0.485614 | Validation loss: 0.630487
Epoch: 96 | Training loss: 0.483736 | Validation loss: 0.628704
Epoch: 97 | Training loss: 0.481881 | Validation loss: 0.626942
Epoch: 98 | Training loss: 0.480047 | Validation loss: 0.625202
Epoch: 99 | Training loss: 0.478236 | Validation loss: 0.623481
Epoch: 100 | Training loss: 0.476446 | Validation loss: 0.621782
Epoch: 101 | Training loss: 0.474677 | Validation loss: 0.620102
Epoch: 102 | Training loss: 0.472928 | Validation loss: 0.618440
Epoch: 103 | Training loss: 0.471199 | Validation loss: 0.616799
```

```
Epoch: 104 | Training loss: 0.469490 | Validation loss: 0.615175
Epoch: 105 | Training loss: 0.467799 | Validation loss: 0.613570
Epoch: 106 | Training loss: 0.466128 | Validation loss: 0.611982
Epoch: 107 | Training loss: 0.464476 | Validation loss: 0.610413
Epoch: 108 | Training loss: 0.462841 | Validation loss: 0.608860
Epoch: 109 | Training loss: 0.461224 | Validation loss: 0.607325
Epoch: 110 | Training loss: 0.459624 | Validation loss: 0.605806
Epoch: 111 | Training loss: 0.458042 | Validation loss: 0.604303
Epoch: 112 | Training loss: 0.456476 | Validation loss: 0.602817
Epoch: 113 | Training loss: 0.454927 | Validation loss: 0.601347
Epoch: 114 | Training loss: 0.453394 | Validation loss: 0.599892
Epoch: 115 | Training loss: 0.451877 | Validation loss: 0.598452
Epoch: 116 | Training loss: 0.450375 | Validation loss: 0.597028
Epoch: 117 | Training loss: 0.448889 | Validation loss: 0.595618
Epoch: 118 | Training loss: 0.447418 | Validation loss: 0.594222
Epoch: 119 | Training loss: 0.445961 | Validation loss: 0.592840
Epoch: 120 | Training loss: 0.444519 | Validation loss: 0.591473
Epoch: 121 | Training loss: 0.443091 | Validation loss: 0.590120
Epoch: 122 | Training loss: 0.441678 | Validation loss: 0.588780
Epoch: 123 | Training loss: 0.440278 | Validation loss: 0.587454
Epoch: 124 | Training loss: 0.438891 | Validation loss: 0.586140
Epoch: 125 | Training loss: 0.437518 | Validation loss: 0.584840
Epoch: 126 | Training loss: 0.436158 | Validation loss: 0.583553
Epoch: 127 | Training loss: 0.434811 | Validation loss: 0.582277
Epoch: 128 | Training loss: 0.433476 | Validation loss: 0.581014
Epoch: 129 | Training loss: 0.432154 | Validation loss: 0.579764
Epoch: 130 | Training loss: 0.430844 | Validation loss: 0.578525
Epoch: 131 | Training loss: 0.429546 | Validation loss: 0.577298
Epoch: 132 | Training loss: 0.428260 | Validation loss: 0.576082
Epoch: 133 | Training loss: 0.426986 | Validation loss: 0.574877
Epoch: 134 | Training loss: 0.425723 | Validation loss: 0.573684
Epoch: 135 | Training loss: 0.424471 | Validation loss: 0.572502
Epoch: 136 | Training loss: 0.423230 | Validation loss: 0.571331
Epoch: 137 | Training loss: 0.422001 | Validation loss: 0.570171
Epoch: 138 | Training loss: 0.420782 | Validation loss: 0.569021
Epoch: 139 | Training loss: 0.419574 | Validation loss: 0.567881
Epoch: 140 | Training loss: 0.418376 | Validation loss: 0.566752
```

```
Epoch: 141 | Training loss: 0.417188 | Validation loss: 0.565632
Epoch: 142 | Training loss: 0.416011 | Validation loss: 0.564523
Epoch: 143 | Training loss: 0.414843 | Validation loss: 0.563423
Epoch: 144 | Training loss: 0.413686 | Validation loss: 0.562333
Epoch: 145 | Training loss: 0.412538 | Validation loss: 0.561254
Epoch: 146 | Training loss: 0.411400 | Validation loss: 0.560182
Epoch: 147 | Training loss: 0.410271 | Validation loss: 0.559120
Epoch: 148 | Training loss: 0.409151 | Validation loss: 0.558067
Epoch: 149 | Training loss: 0.408040 | Validation loss: 0.557023
Epoch: 150 | Training loss: 0.406939 | Validation loss: 0.555988
Epoch: 151 | Training loss: 0.405846 | Validation loss: 0.554962
Epoch: 152 | Training loss: 0.404762 | Validation loss: 0.553944
Epoch: 153 | Training loss: 0.403687 | Validation loss: 0.552935
Epoch: 154 | Training loss: 0.402620 | Validation loss: 0.551934
Epoch: 155 | Training loss: 0.401562 | Validation loss: 0.550942
lr = 0.000100, alpha= 0.001000; Predict_Class:0.823333
Epoch: 0 | Training loss: 1.084229 | Validation loss: 1.083282
Epoch: 1 | Training loss: 1.056039 | Validation loss: 1.069048
Epoch: 2 | Training loss: 1.030950 | Validation loss: 1.055704
Epoch: 3 | Training loss: 1.008289 | Validation loss: 1.043107
Epoch: 4 | Training loss: 0.987562 | Validation loss: 1.031138
Epoch: 5 | Training loss: 0.968411 | Validation loss: 1.019712
Epoch: 6 | Training loss: 0.950576 | Validation loss: 1.008768
Epoch: 7 | Training loss: 0.933872 | Validation loss: 0.998257
Epoch: 8 | Training loss: 0.918157 | Validation loss: 0.988137
Epoch: 9 | Training loss: 0.903311 | Validation loss: 0.978375
Epoch: 10 | Training loss: 0.889243 | Validation loss: 0.968946
Epoch: 11 | Training loss: 0.875881 | Validation loss: 0.959830
Epoch: 12 | Training loss: 0.863165 | Validation loss: 0.951006
Epoch: 13 | Training loss: 0.851039 | Validation loss: 0.942461
Epoch: 14 | Training loss: 0.839455 | Validation loss: 0.934177
Epoch: 15 | Training loss: 0.828372 | Validation loss: 0.926142
Epoch: 16 | Training loss: 0.817757 | Validation loss: 0.918344
Epoch: 17 | Training loss: 0.807578 | Validation loss: 0.910770
Epoch: 18 | Training loss: 0.797802 | Validation loss: 0.903412
Epoch: 19 | Training loss: 0.788404 | Validation loss: 0.896258
Epoch: 20 | Training loss: 0.779363 | Validation loss: 0.889301
```

```
Epoch: 21 | Training loss: 0.770653 | Validation loss: 0.882531
Epoch: 22 | Training loss: 0.762256 | Validation loss: 0.875942
Epoch: 23 | Training loss: 0.754155 | Validation loss: 0.869526
Epoch: 24 | Training loss: 0.746330 | Validation loss: 0.863274
Epoch: 25 | Training loss: 0.738767 | Validation loss: 0.857183
Epoch: 26 | Training loss: 0.731452 | Validation loss: 0.851245
Epoch: 27 | Training loss: 0.724371 | Validation loss: 0.845452
Epoch: 28 | Training loss: 0.717510 | Validation loss: 0.839800
Epoch: 29 | Training loss: 0.710861 | Validation loss: 0.834286
Epoch: 30 | Training loss: 0.704413 | Validation loss: 0.828903
Epoch: 31 | Training loss: 0.698154 | Validation loss: 0.823643
Epoch: 32 | Training loss: 0.692075 | Validation loss: 0.818507
Epoch: 33 | Training loss: 0.686168 | Validation loss: 0.813488
Epoch: 34 | Training loss: 0.680425 | Validation loss: 0.808583
Epoch: 35 | Training loss: 0.674838 | Validation loss: 0.803786
Epoch: 36 | Training loss: 0.669401 | Validation loss: 0.799093
Epoch: 37 | Training loss: 0.664105 | Validation loss: 0.794503
Epoch: 38 | Training loss: 0.658946 | Validation loss: 0.790010
Epoch: 39 | Training loss: 0.653917 | Validation loss: 0.785613
Epoch: 40 | Training loss: 0.649013 | Validation loss: 0.781307
Epoch: 41 | Training loss: 0.644228 | Validation loss: 0.777091
Epoch: 42 | Training loss: 0.639559 | Validation loss: 0.772960
Epoch: 43 | Training loss: 0.634999 | Validation loss: 0.768912
Epoch: 44 | Training loss: 0.630545 | Validation loss: 0.764945
Epoch: 45 | Training loss: 0.626193 | Validation loss: 0.761056
Epoch: 46 | Training loss: 0.621939 | Validation loss: 0.757242
Epoch: 47 | Training loss: 0.617778 | Validation loss: 0.753501
Epoch: 48 | Training loss: 0.613707 | Validation loss: 0.749831
Epoch: 49 | Training loss: 0.609724 | Validation loss: 0.746230
Epoch: 50 | Training loss: 0.605824 | Validation loss: 0.742696
Epoch: 51 | Training loss: 0.602006 | Validation loss: 0.739226
Epoch: 52 | Training loss: 0.598265 | Validation loss: 0.735820
Epoch: 53 | Training loss: 0.594600 | Validation loss: 0.732474
Epoch: 54 | Training loss: 0.591007 | Validation loss: 0.729187
Epoch: 55 | Training loss: 0.587485 | Validation loss: 0.725958
Epoch: 56 | Training loss: 0.584031 | Validation loss: 0.722785
Epoch: 57 | Training loss: 0.580642 | Validation loss: 0.719667
```

```
Epoch: 58 | Training loss: 0.577316 | Validation loss: 0.716601
Epoch: 59 | Training loss: 0.574052 | Validation loss: 0.713587
Epoch: 60 | Training loss: 0.570848 | Validation loss: 0.710623
Epoch: 61 | Training loss: 0.567701 | Validation loss: 0.707708
Epoch: 62 | Training loss: 0.564610 | Validation loss: 0.704840
Epoch: 63 | Training loss: 0.561574 | Validation loss: 0.702018
Epoch: 64 | Training loss: 0.558590 | Validation loss: 0.699242
Epoch: 65 | Training loss: 0.555656 | Validation loss: 0.696509
Epoch: 66 | Training loss: 0.552772 | Validation loss: 0.693819
Epoch: 67 | Training loss: 0.549937 | Validation loss: 0.691171
Epoch: 68 | Training loss: 0.547148 | Validation loss: 0.688563
Epoch: 69 | Training loss: 0.544403 | Validation loss: 0.685995
Epoch: 70 | Training loss: 0.541703 | Validation loss: 0.683465
Epoch: 71 | Training loss: 0.539046 | Validation loss: 0.680972
Epoch: 72 | Training loss: 0.536430 | Validation loss: 0.678517
Epoch: 73 | Training loss: 0.533855 | Validation loss: 0.676097
Epoch: 74 | Training loss: 0.531320 | Validation loss: 0.673713
Epoch: 75 | Training loss: 0.528822 | Validation loss: 0.671363
Epoch: 76 | Training loss: 0.526362 | Validation loss: 0.669047
Epoch: 77 | Training loss: 0.523939 | Validation loss: 0.666763
Epoch: 78 | Training loss: 0.521551 | Validation loss: 0.664511
Epoch: 79 | Training loss: 0.519197 | Validation loss: 0.662289
Epoch: 80 | Training loss: 0.516877 | Validation loss: 0.660099
Epoch: 81 | Training loss: 0.514590 | Validation loss: 0.657939
Epoch: 82 | Training loss: 0.512335 | Validation loss: 0.655807
Epoch: 83 | Training loss: 0.510111 | Validation loss: 0.653704
Epoch: 84 | Training loss: 0.507918 | Validation loss: 0.651629
Epoch: 85 | Training loss: 0.505755 | Validation loss: 0.649581
Epoch: 86 | Training loss: 0.503620 | Validation loss: 0.647561
Epoch: 87 | Training loss: 0.501514 | Validation loss: 0.645566
Epoch: 88 | Training loss: 0.499435 | Validation loss: 0.643596
Epoch: 89 | Training loss: 0.497384 | Validation loss: 0.641652
Epoch: 90 | Training loss: 0.495359 | Validation loss: 0.639733
Epoch: 91 | Training loss: 0.493361 | Validation loss: 0.637837
Epoch: 92 | Training loss: 0.491387 | Validation loss: 0.635966
Epoch: 93 | Training loss: 0.489438 | Validation loss: 0.634117
Epoch: 94 | Training loss: 0.487513 | Validation loss: 0.632290
```

```
Epoch: 95 | Training loss: 0.485612 | Validation loss: 0.630486
Epoch: 96 | Training loss: 0.483734 | Validation loss: 0.628704
Epoch: 97 | Training loss: 0.481879 | Validation loss: 0.626942
Epoch: 98 | Training loss: 0.480046 | Validation loss: 0.625201
Epoch: 99 | Training loss: 0.478235 | Validation loss: 0.623481
Epoch: 100 | Training loss: 0.476444 | Validation loss: 0.621781
Epoch: 101 | Training loss: 0.474675 | Validation loss: 0.620100
Epoch: 102 | Training loss: 0.472926 | Validation loss: 0.618440
Epoch: 103 | Training loss: 0.471197 | Validation loss: 0.616798
Epoch: 104 | Training loss: 0.469488 | Validation loss: 0.615175
Epoch: 105 | Training loss: 0.467798 | Validation loss: 0.613570
Epoch: 106 | Training loss: 0.466127 | Validation loss: 0.611982
Epoch: 107 | Training loss: 0.464474 | Validation loss: 0.610412
Epoch: 108 | Training loss: 0.462840 | Validation loss: 0.608860
Epoch: 109 | Training loss: 0.461223 | Validation loss: 0.607324
Epoch: 110 | Training loss: 0.459623 | Validation loss: 0.605806
Epoch: 111 | Training loss: 0.458041 | Validation loss: 0.604303
Epoch: 112 | Training loss: 0.456475 | Validation loss: 0.602817
Epoch: 113 | Training loss: 0.454926 | Validation loss: 0.601347
Epoch: 114 | Training loss: 0.453393 | Validation loss: 0.599892
Epoch: 115 | Training loss: 0.451876 | Validation loss: 0.598452
Epoch: 116 | Training loss: 0.450374 | Validation loss: 0.597027
Epoch: 117 | Training loss: 0.448888 | Validation loss: 0.595617
Epoch: 118 | Training loss: 0.447417 | Validation loss: 0.594222
Epoch: 119 | Training loss: 0.445960 | Validation loss: 0.592841
Epoch: 120 | Training loss: 0.444518 | Validation loss: 0.591473
Epoch: 121 | Training loss: 0.443091 | Validation loss: 0.590120
Epoch: 122 | Training loss: 0.441677 | Validation loss: 0.588780
Epoch: 123 | Training loss: 0.440277 | Validation loss: 0.587454
Epoch: 124 | Training loss: 0.438890 | Validation loss: 0.586140
Epoch: 125 | Training loss: 0.437517 | Validation loss: 0.584840
Epoch: 126 | Training loss: 0.436157 | Validation loss: 0.583553
Epoch: 127 | Training loss: 0.434810 | Validation loss: 0.582278
Epoch: 128 | Training loss: 0.433475 | Validation loss: 0.581015
Epoch: 129 | Training loss: 0.432153 | Validation loss: 0.579764
Epoch: 130 | Training loss: 0.430843 | Validation loss: 0.578525
Epoch: 131 | Training loss: 0.429545 | Validation loss: 0.577298
```

```
Epoch: 132 | Training loss: 0.428259 | Validation loss: 0.576082
Epoch: 133 | Training loss: 0.426985 | Validation loss: 0.574878
Epoch: 134 | Training loss: 0.425722 | Validation loss: 0.573685
Epoch: 135 | Training loss: 0.424470 | Validation loss: 0.572503
Epoch: 136 | Training loss: 0.423230 | Validation loss: 0.571332
Epoch: 137 | Training loss: 0.422000 | Validation loss: 0.570171
Epoch: 138 | Training loss: 0.420781 | Validation loss: 0.569022
Epoch: 139 | Training loss: 0.419573 | Validation loss: 0.567882
Epoch: 140 | Training loss: 0.418375 | Validation loss: 0.566753
Epoch: 141 | Training loss: 0.417188 | Validation loss: 0.565634
Epoch: 142 | Training loss: 0.416010 | Validation loss: 0.564524
Epoch: 143 | Training loss: 0.414843 | Validation loss: 0.563425
Epoch: 144 | Training loss: 0.413686 | Validation loss: 0.562335
Epoch: 145 | Training loss: 0.412538 | Validation loss: 0.561254
Epoch: 146 | Training loss: 0.411399 | Validation loss: 0.560183
Epoch: 147 | Training loss: 0.410270 | Validation loss: 0.559121
Epoch: 148 | Training loss: 0.409151 | Validation loss: 0.558068
Epoch: 149 | Training loss: 0.408040 | Validation loss: 0.557025
Epoch: 150 | Training loss: 0.406939 | Validation loss: 0.555990
Epoch: 151 | Training loss: 0.405846 | Validation loss: 0.554964
Epoch: 152 | Training loss: 0.404762 | Validation loss: 0.553946
Epoch: 153 | Training loss: 0.403687 | Validation loss: 0.552937
Epoch: 154 | Training loss: 0.402620 | Validation loss: 0.551936
Epoch: 155 | Training loss: 0.401562 | Validation loss: 0.550944
lr = 0.000100, alpha= 0.000500; Predict_Class:0.823333
Epoch: 0 | Training loss: 1.084245 | Validation loss: 1.083293
Epoch: 1 | Training loss: 1.056046 | Validation loss: 1.069047
Epoch: 2 | Training loss: 1.030937 | Validation loss: 1.055701
Epoch: 3 | Training loss: 1.008274 | Validation loss: 1.043103
Epoch: 4 | Training loss: 0.987548 | Validation loss: 1.031136
Epoch: 5 | Training loss: 0.968402 | Validation loss: 1.019712
Epoch: 6 | Training loss: 0.950574 | Validation loss: 1.008768
Epoch: 7 | Training loss: 0.933872 | Validation loss: 0.998254
Epoch: 8 | Training loss: 0.918150 | Validation loss: 0.988131
Epoch: 9 | Training loss: 0.903303 | Validation loss: 0.978367
Epoch: 10 | Training loss: 0.889236 | Validation loss: 0.968939
Epoch: 11 | Training loss: 0.875876 | Validation loss: 0.959825
```

```
Epoch: 12 | Training loss: 0.863162 | Validation loss: 0.951001
Epoch: 13 | Training loss: 0.851036 | Validation loss: 0.942456
Epoch: 14 | Training loss: 0.839454 | Validation loss: 0.934172
Epoch: 15 | Training loss: 0.828374 | Validation loss: 0.926136
Epoch: 16 | Training loss: 0.817759 | Validation loss: 0.918340
Epoch: 17 | Training loss: 0.807580 | Validation loss: 0.910768
Epoch: 18 | Training loss: 0.797805 | Validation loss: 0.903408
Epoch: 19 | Training loss: 0.788408 | Validation loss: 0.896255
Epoch: 20 | Training loss: 0.779367 | Validation loss: 0.889298
Epoch: 21 | Training loss: 0.770659 | Validation loss: 0.882529
Epoch: 22 | Training loss: 0.762264 | Validation loss: 0.875939
Epoch: 23 | Training loss: 0.754162 | Validation loss: 0.869521
Epoch: 24 | Training loss: 0.746337 | Validation loss: 0.863271
Epoch: 25 | Training loss: 0.738775 | Validation loss: 0.857181
Epoch: 26 | Training loss: 0.731460 | Validation loss: 0.851241
Epoch: 27 | Training loss: 0.724379 | Validation loss: 0.845449
Epoch: 28 | Training loss: 0.717519 | Validation loss: 0.839798
Epoch: 29 | Training loss: 0.710871 | Validation loss: 0.834283
Epoch: 30 | Training loss: 0.704422 | Validation loss: 0.828900
Epoch: 31 | Training loss: 0.698161 | Validation loss: 0.823642
Epoch: 32 | Training loss: 0.692082 | Validation loss: 0.818506
Epoch: 33 | Training loss: 0.686176 | Validation loss: 0.813487
Epoch: 34 | Training loss: 0.680432 | Validation loss: 0.808580
Epoch: 35 | Training loss: 0.674845 | Validation loss: 0.803785
Epoch: 36 | Training loss: 0.669407 | Validation loss: 0.799093
Epoch: 37 | Training loss: 0.664112 | Validation loss: 0.794504
Epoch: 38 | Training loss: 0.658953 | Validation loss: 0.790011
Epoch: 39 | Training loss: 0.653924 | Validation loss: 0.785614
Epoch: 40 | Training loss: 0.649020 | Validation loss: 0.781308
Epoch: 41 | Training loss: 0.644235 | Validation loss: 0.777091
Epoch: 42 | Training loss: 0.639565 | Validation loss: 0.772960
Epoch: 43 | Training loss: 0.635005 | Validation loss: 0.768913
Epoch: 44 | Training loss: 0.630551 | Validation loss: 0.764945
Epoch: 45 | Training loss: 0.626198 | Validation loss: 0.761055
Epoch: 46 | Training loss: 0.621943 | Validation loss: 0.757240
Epoch: 47 | Training loss: 0.617781 | Validation loss: 0.753500
Epoch: 48 | Training loss: 0.613711 | Validation loss: 0.749830
```

```
Epoch: 49 | Training loss: 0.609727 | Validation loss: 0.746229
Epoch: 50 | Training loss: 0.605828 | Validation loss: 0.742695
Epoch: 51 | Training loss: 0.602009 | Validation loss: 0.739225
Epoch: 52 | Training loss: 0.598269 | Validation loss: 0.735819
Epoch: 53 | Training loss: 0.594603 | Validation loss: 0.732473
Epoch: 54 | Training loss: 0.591010 | Validation loss: 0.729187
Epoch: 55 | Training loss: 0.587488 | Validation loss: 0.725959
Epoch: 56 | Training loss: 0.584033 | Validation loss: 0.722786
Epoch: 57 | Training loss: 0.580644 | Validation loss: 0.719667
Epoch: 58 | Training loss: 0.577319 | Validation loss: 0.716602
Epoch: 59 | Training loss: 0.574055 | Validation loss: 0.713587
Epoch: 60 | Training loss: 0.570850 | Validation loss: 0.710623
Epoch: 61 | Training loss: 0.567703 | Validation loss: 0.707707
Epoch: 62 | Training loss: 0.564612 | Validation loss: 0.704839
Epoch: 63 | Training loss: 0.561575 | Validation loss: 0.702018
Epoch: 64 | Training loss: 0.558590 | Validation loss: 0.699241
Epoch: 65 | Training loss: 0.555657 | Validation loss: 0.696508
Epoch: 66 | Training loss: 0.552773 | Validation loss: 0.693818
Epoch: 67 | Training loss: 0.549937 | Validation loss: 0.691170
Epoch: 68 | Training loss: 0.547148 | Validation loss: 0.688562
Epoch: 69 | Training loss: 0.544404 | Validation loss: 0.685994
Epoch: 70 | Training loss: 0.541704 | Validation loss: 0.683463
Epoch: 71 | Training loss: 0.539046 | Validation loss: 0.680971
Epoch: 72 | Training loss: 0.536431 | Validation loss: 0.678516
Epoch: 73 | Training loss: 0.533855 | Validation loss: 0.676096
Epoch: 74 | Training loss: 0.531320 | Validation loss: 0.673712
Epoch: 75 | Training loss: 0.528823 | Validation loss: 0.671362
Epoch: 76 | Training loss: 0.526363 | Validation loss: 0.669045
Epoch: 77 | Training loss: 0.523939 | Validation loss: 0.666761
Epoch: 78 | Training loss: 0.521551 | Validation loss: 0.664510
Epoch: 79 | Training loss: 0.519197 | Validation loss: 0.662289
Epoch: 80 | Training loss: 0.516877 | Validation loss: 0.660099
Epoch: 81 | Training loss: 0.514590 | Validation loss: 0.657938
Epoch: 82 | Training loss: 0.512335 | Validation loss: 0.655807
Epoch: 83 | Training loss: 0.510112 | Validation loss: 0.653703
Epoch: 84 | Training loss: 0.507918 | Validation loss: 0.651628
Epoch: 85 | Training loss: 0.505755 | Validation loss: 0.649581
```

```
Epoch: 86 | Training loss: 0.503620 | Validation loss: 0.647560
Epoch: 87 | Training loss: 0.501514 | Validation loss: 0.645565
Epoch: 88 | Training loss: 0.499436 | Validation loss: 0.643596
Epoch: 89 | Training loss: 0.497385 | Validation loss: 0.641651
Epoch: 90 | Training loss: 0.495360 | Validation loss: 0.639732
Epoch: 91 | Training loss: 0.493361 | Validation loss: 0.637837
Epoch: 92 | Training loss: 0.491388 | Validation loss: 0.635965
Epoch: 93 | Training loss: 0.489439 | Validation loss: 0.634116
Epoch: 94 | Training loss: 0.487514 | Validation loss: 0.632290
Epoch: 95 | Training loss: 0.485613 | Validation loss: 0.630485
Epoch: 96 | Training loss: 0.483735 | Validation loss: 0.628702
Epoch: 97 | Training loss: 0.481880 | Validation loss: 0.626941
Epoch: 98 | Training loss: 0.480046 | Validation loss: 0.625200
Epoch: 99 | Training loss: 0.478235 | Validation loss: 0.623480
Epoch: 100 | Training loss: 0.476445 | Validation loss: 0.621780
Epoch: 101 | Training loss: 0.474676 | Validation loss: 0.620100
Epoch: 102 | Training loss: 0.472927 | Validation loss: 0.618439
Epoch: 103 | Training loss: 0.471198 | Validation loss: 0.616797
Epoch: 104 | Training loss: 0.469489 | Validation loss: 0.615174
Epoch: 105 | Training loss: 0.467798 | Validation loss: 0.613568
Epoch: 106 | Training loss: 0.466127 | Validation loss: 0.611981
Epoch: 107 | Training loss: 0.464474 | Validation loss: 0.610411
Epoch: 108 | Training loss: 0.462840 | Validation loss: 0.608859
Epoch: 109 | Training loss: 0.461223 | Validation loss: 0.607324
Epoch: 110 | Training loss: 0.459623 | Validation loss: 0.605805
Epoch: 111 | Training loss: 0.458041 | Validation loss: 0.604303
Epoch: 112 | Training loss: 0.456475 | Validation loss: 0.602818
Epoch: 113 | Training loss: 0.454926 | Validation loss: 0.601347
Epoch: 114 | Training loss: 0.453393 | Validation loss: 0.599892
Epoch: 115 | Training loss: 0.451876 | Validation loss: 0.598452
Epoch: 116 | Training loss: 0.450375 | Validation loss: 0.597028
Epoch: 117 | Training loss: 0.448888 | Validation loss: 0.595617
Epoch: 118 | Training loss: 0.447417 | Validation loss: 0.594222
Epoch: 119 | Training loss: 0.445961 | Validation loss: 0.592841
Epoch: 120 | Training loss: 0.444519 | Validation loss: 0.591474
Epoch: 121 | Training loss: 0.443091 | Validation loss: 0.590120
Epoch: 122 | Training loss: 0.441677 | Validation loss: 0.588781
```

```
Epoch: 123 | Training loss: 0.440277 | Validation loss: 0.587454
Epoch: 124 | Training loss: 0.438890 | Validation loss: 0.586141
Epoch: 125 | Training loss: 0.437517 | Validation loss: 0.584841
Epoch: 126 | Training loss: 0.436157 | Validation loss: 0.583553
Epoch: 127 | Training loss: 0.434810 | Validation loss: 0.582278
Epoch: 128 | Training loss: 0.433475 | Validation loss: 0.581015
Epoch: 129 | Training loss: 0.432153 | Validation loss: 0.579764
Epoch: 130 | Training loss: 0.430843 | Validation loss: 0.578525
Epoch: 131 | Training loss: 0.429545 | Validation loss: 0.577297
Epoch: 132 | Training loss: 0.428259 | Validation loss: 0.576082
Epoch: 133 | Training loss: 0.426985 | Validation loss: 0.574878
Epoch: 134 | Training loss: 0.425722 | Validation loss: 0.573685
Epoch: 135 | Training loss: 0.424470 | Validation loss: 0.572503
Epoch: 136 | Training loss: 0.423229 | Validation loss: 0.571332
Epoch: 137 | Training loss: 0.422000 | Validation loss: 0.570171
Epoch: 138 | Training loss: 0.420781 | Validation loss: 0.569021
Epoch: 139 | Training loss: 0.419573 | Validation loss: 0.567881
Epoch: 140 | Training loss: 0.418375 | Validation loss: 0.566752
Epoch: 141 | Training loss: 0.417188 | Validation loss: 0.565632
Epoch: 142 | Training loss: 0.416010 | Validation loss: 0.564523
Epoch: 143 | Training loss: 0.414843 | Validation loss: 0.563424
Epoch: 144 | Training loss: 0.413685 | Validation loss: 0.562334
Epoch: 145 | Training loss: 0.412537 | Validation loss: 0.561253
Epoch: 146 | Training loss: 0.411399 | Validation loss: 0.560183
Epoch: 147 | Training loss: 0.410270 | Validation loss: 0.559120
Epoch: 148 | Training loss: 0.409150 | Validation loss: 0.558067
Epoch: 149 | Training loss: 0.408040 | Validation loss: 0.557024
Epoch: 150 | Training loss: 0.406938 | Validation loss: 0.555989
Epoch: 151 | Training loss: 0.405846 | Validation loss: 0.554963
Epoch: 152 | Training loss: 0.404762 | Validation loss: 0.553946
Epoch: 153 | Training loss: 0.403687 | Validation loss: 0.552937
Epoch: 154 | Training loss: 0.402620 | Validation loss: 0.551936
Epoch: 155 | Training loss: 0.401562 | Validation loss: 0.550943
lr = 0.000100, alpha= 0.000100; Predict_Class:0.823333
```

From the previous result, we can see that as decreasing the value of lr and alpha, the F1-Score is increasing.

In conclusion,the both lr and alpha has the great impact the training model.

### 4.0.2 Now evaluate BOW-tfidf…

[63]:

```
w_count, loss_tr_count, dev_loss_count = SGD(X_tr_tfidf, data_tr_label,vocab,
                                            X_dev=X_dev_tfidf,
                                            Y_dev=data_dev_label,
                                            num_classes=3,
                                            lr=0.0001,
                                            alpha=0.001,
                                            epochs=200)
```

```
Epoch: 0 | Training loss: 0.959970 | Validation loss: 0.957439
Epoch: 1 | Training loss: 0.763611 | Validation loss: 0.867650
Epoch: 2 | Training loss: 0.650473 | Validation loss: 0.803574
Epoch: 3 | Training loss: 0.574929 | Validation loss: 0.754995
Epoch: 4 | Training loss: 0.520227 | Validation loss: 0.716595
Epoch: 5 | Training loss: 0.478179 | Validation loss: 0.685185
Epoch: 6 | Training loss: 0.444663 | Validation loss: 0.659090
Epoch: 7 | Training loss: 0.417000 | Validation loss: 0.636840
Epoch: 8 | Training loss: 0.393671 | Validation loss: 0.617503
Epoch: 9 | Training loss: 0.373643 | Validation loss: 0.600592
Epoch: 10 | Training loss: 0.356148 | Validation loss: 0.585546
Epoch: 11 | Training loss: 0.340754 | Validation loss: 0.572148
Epoch: 12 | Training loss: 0.327017 | Validation loss: 0.560089
Epoch: 13 | Training loss: 0.314632 | Validation loss: 0.549119
Epoch: 14 | Training loss: 0.303426 | Validation loss: 0.539062
Epoch: 15 | Training loss: 0.293215 | Validation loss: 0.529844
Epoch: 16 | Training loss: 0.283825 | Validation loss: 0.521364
Epoch: 17 | Training loss: 0.275171 | Validation loss: 0.513474
Epoch: 18 | Training loss: 0.267169 | Validation loss: 0.506176
Epoch: 19 | Training loss: 0.259722 | Validation loss: 0.499357
Epoch: 20 | Training loss: 0.252761 | Validation loss: 0.492999
Epoch: 21 | Training loss: 0.246258 | Validation loss: 0.487005
Epoch: 22 | Training loss: 0.240162 | Validation loss: 0.481363
Epoch: 23 | Training loss: 0.234407 | Validation loss: 0.476022
Epoch: 24 | Training loss: 0.228987 | Validation loss: 0.471008
```

```
Epoch: 25 | Training loss: 0.223852 | Validation loss: 0.466253
Epoch: 26 | Training loss: 0.218994 | Validation loss: 0.461734
Epoch: 27 | Training loss: 0.214367 | Validation loss: 0.457388
Epoch: 28 | Training loss: 0.209991 | Validation loss: 0.453303
Epoch: 29 | Training loss: 0.205807 | Validation loss: 0.449379
Epoch: 30 | Training loss: 0.201817 | Validation loss: 0.445660
Epoch: 31 | Training loss: 0.198002 | Validation loss: 0.442101
Epoch: 32 | Training loss: 0.194354 | Validation loss: 0.438700
Epoch: 33 | Training loss: 0.190864 | Validation loss: 0.435394
Epoch: 34 | Training loss: 0.187513 | Validation loss: 0.432245
Epoch: 35 | Training loss: 0.184288 | Validation loss: 0.429241
Epoch: 36 | Training loss: 0.181199 | Validation loss: 0.426325
Epoch: 37 | Training loss: 0.178220 | Validation loss: 0.423520
Epoch: 38 | Training loss: 0.175357 | Validation loss: 0.420853
Epoch: 39 | Training loss: 0.172589 | Validation loss: 0.418254
Epoch: 40 | Training loss: 0.169925 | Validation loss: 0.415731
Epoch: 41 | Training loss: 0.167355 | Validation loss: 0.413295
Epoch: 42 | Training loss: 0.164866 | Validation loss: 0.410973
Epoch: 43 | Training loss: 0.162458 | Validation loss: 0.408677
Epoch: 44 | Training loss: 0.160134 | Validation loss: 0.406480
Epoch: 45 | Training loss: 0.157879 | Validation loss: 0.404358
Epoch: 46 | Training loss: 0.155694 | Validation loss: 0.402284
Epoch: 47 | Training loss: 0.153574 | Validation loss: 0.400257
Epoch: 48 | Training loss: 0.151522 | Validation loss: 0.398323
Epoch: 49 | Training loss: 0.149530 | Validation loss: 0.396438
Epoch: 50 | Training loss: 0.147595 | Validation loss: 0.394620
Epoch: 51 | Training loss: 0.145705 | Validation loss: 0.392820
Epoch: 52 | Training loss: 0.143878 | Validation loss: 0.391112
Epoch: 53 | Training loss: 0.142105 | Validation loss: 0.389427
Epoch: 54 | Training loss: 0.140370 | Validation loss: 0.387795
Epoch: 55 | Training loss: 0.138685 | Validation loss: 0.386205
Epoch: 56 | Training loss: 0.137041 | Validation loss: 0.384659
Epoch: 57 | Training loss: 0.135442 | Validation loss: 0.383138
Epoch: 58 | Training loss: 0.133885 | Validation loss: 0.381675
Epoch: 59 | Training loss: 0.132362 | Validation loss: 0.380233
Epoch: 60 | Training loss: 0.130876 | Validation loss: 0.378855
Epoch: 61 | Training loss: 0.129429 | Validation loss: 0.377465
```

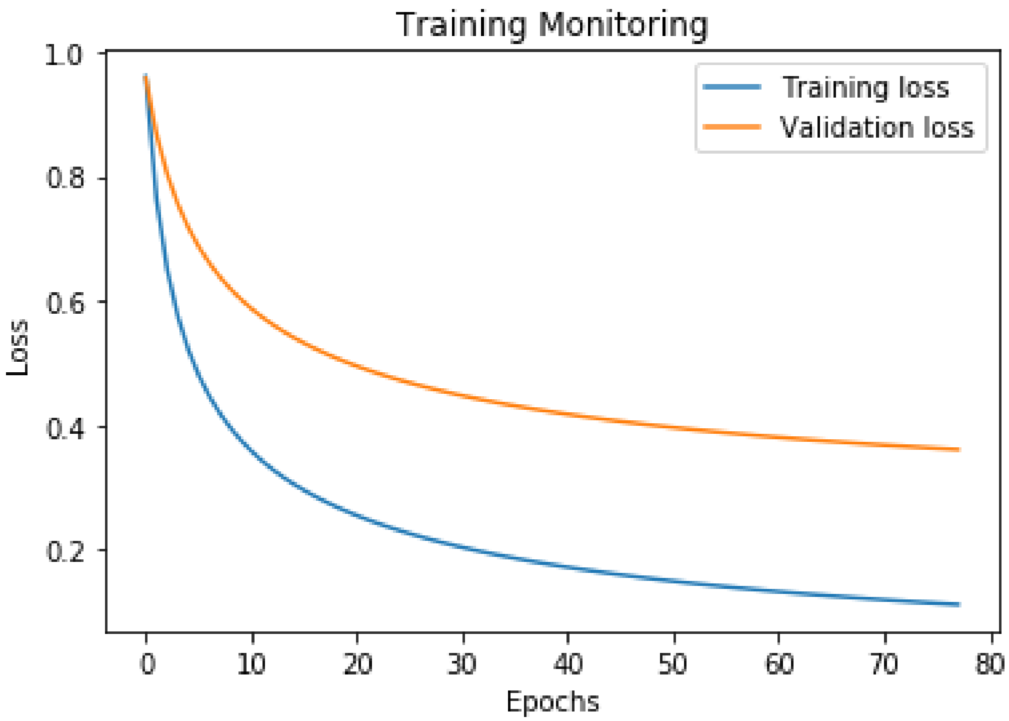
```
Epoch: 62 | Training loss: 0.128018 | Validation loss: 0.376132
Epoch: 63 | Training loss: 0.126639 | Validation loss: 0.374839
Epoch: 64 | Training loss: 0.125284 | Validation loss: 0.373554
Epoch: 65 | Training loss: 0.123971 | Validation loss: 0.372315
Epoch: 66 | Training loss: 0.122680 | Validation loss: 0.371096
Epoch: 67 | Training loss: 0.121423 | Validation loss: 0.369901
Epoch: 68 | Training loss: 0.120193 | Validation loss: 0.368746
Epoch: 69 | Training loss: 0.118986 | Validation loss: 0.367622
Epoch: 70 | Training loss: 0.117807 | Validation loss: 0.366534
Epoch: 71 | Training loss: 0.116650 | Validation loss: 0.365446
Epoch: 72 | Training loss: 0.115521 | Validation loss: 0.364399
Epoch: 73 | Training loss: 0.114408 | Validation loss: 0.363356
Epoch: 74 | Training loss: 0.113325 | Validation loss: 0.362338
Epoch: 75 | Training loss: 0.112265 | Validation loss: 0.361331
Epoch: 76 | Training loss: 0.111224 | Validation loss: 0.360346
```

[64]:

```python
x = np.linspace(0,len(loss_tr_count),len(loss_tr_count))
y1, y2 = loss_tr_count, dev_loss_count

plt.plot(x, y1,label='Training loss')
plt.plot(x, y2, label='Validation loss')

plt.title('Training Monitoring')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



The model is overfit

[65]:

```
preds_te_count = predict_class(X = X_test_count, weights = w_count)
Y_te = data_test_label

print('Accuracy:', accuracy_score(Y_te,preds_te_count))
print('Precision:', precision_score(Y_te,preds_te_count))
print('Recall:', recall_score(Y_te,preds_te_count))
print('F1-Score:', f1_score(Y_te,preds_te_count))
```

```
Accuracy: 0.8755555555555555
Precision: 0.8417508417508418
Recall: 0.8333333333333334
F1-Score: 0.8375209380234506
```

## 4.1 Full Results

Add here your results:

| LR | Precision | Recall | F1-Score |
|---|---|---|---|
| BOW-count | 0.8233 | 0.8233 | 0.8233 |
| BOW-tfidf | 0.8417 | 0.8333 | 0.8375 |